3D CT DATA SEGMENTATION

LAI CHEW PING
WEK020088

Under the supervision of
Ms. Mangalam Sankupellay

Moderator:
Prof. Dr. Ir. Selvanathan Narainasamy

This project is submitted to the
Faculty of Computer Science and Information Technology
University of Malaya
in partial fulfillment of the requirement for the degree of
Bachelor of Computer Science

Session 2004/2005

ABSTRACT

The purpose of this project is to explore, understand and utilize a three-dimensional (3D) visualization and segmentation software to perform segmentation on 3D Computed Tomography (CT) data sets. The software selected in this project is Amira developer version 3.1. A step by step guide on how segmentation is done using Amira is shown.

This report contains six chapters which uncovered the details of the project from the initial planning to analysis and design as well as implementation at the end of the project. It includes the selection of development tools, methodology, system requirements and hierarchy. Finally, the discussions on problems encountered and recommended solutions, system strength and limitation, future enhancements and conclusion are provided.

ACKNOWLEDGEMENT

First, I would like to express my gratitude to Ms. Mangalam Sankupelley for her supervision and direction through out the project. She had given me invaluable guidance and advice towards the accomplishment of the project. Also, not forgetting to forward my appreciation to Prof. Dr. Ir. Selvanathan Narainasamy for being the moderator of this project.

Next, I am truly thankful to have a team of course mates who had given me their encouragement and cooperation to making this project a success. Without being self-centered, they are keen to teach and share the knowledge they have.

Last but not least, to my family. I am grateful for their endless care and support all these years. They had given me the strength to carry on and never give up.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
ANN	Artificial Neural Network
CT	Computed Tomography
MRF	Markov Random Field
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography
ROI	Region of Interest
SE	Structural Element

CHAPTER 1

INTRODUCTION

CHAPTER 1 – INTRODUCTION

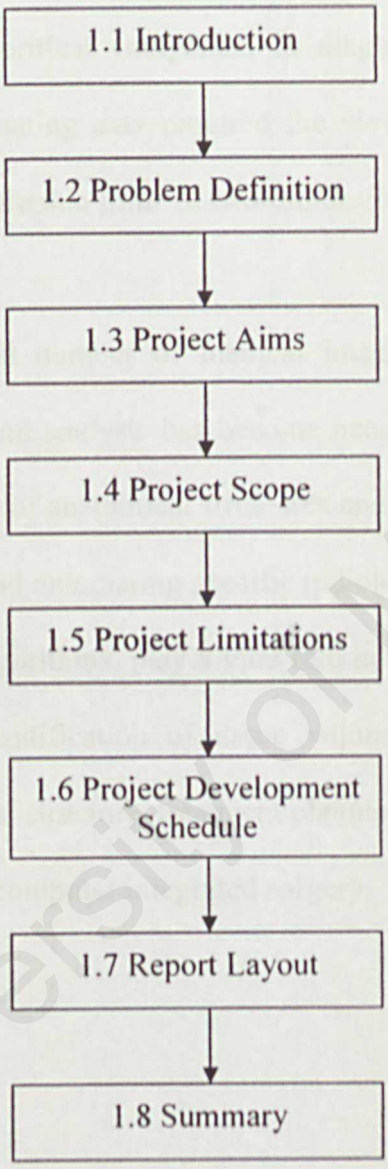


Figure 1.1: Overview of Chapter 1

1.1 Introduction

Diagnostic imaging is an invaluable tool in medicine today. Magnetic resonance imaging (MRI), computed tomography (CT), digital mammography, and other imaging modalities

provide an effective means for non-invasively mapping the anatomy of a subject. These technologies have greatly increased knowledge of normal and diseased anatomy for medical research and are a critical component in diagnosis and treatment planning. Traditional methods for evaluating data required the viewer to mentally reconstruct a volume from a series of radiographic films or two-dimensional images.

With the increasing size and number of medical images, the use of computers in facilitating their processing and analysis has become necessary. In particular, computer algorithms for the delineation of anatomical structures and other regions of interest are a key component in assisting and automating specific radiological tasks. These algorithms, called image segmentation algorithms, play a vital role in numerous biomedical imaging applications such as the quantification of tissue volumes, diagnosis, localization of pathology, study of anatomical structure, treatment planning, partial volume correction of functional imaging data, and computer integrated surgery.

1.2 Problem Definition

The amount and diversity of three-dimensional (3D) image data being studied in structural biology has increased steadily with advances in microscope and computer technology. Currently, there is a high demand for computer software that can analyze and display such complex data sets. In addition to more elaborate and realistic radio-biological models, what will be needed in the future to make this biological treatment planning a routine tool are procedures for individual three-dimensional image

segmentation for the definition of critical organs or even functional sub-units of critical organs.

Classically, image segmentation is defined as the partitioning of an image into non-overlapping, constituent regions which are homogeneous with respect to some characteristic such as intensity or texture. Errors, inaccuracies, and oversights that are made during the first step of image segmentation dramatically influence the effectiveness of treatment planning and the outcome of the whole radio-therapeutic process. This shows the importance of computer technology and software tools that enable the surgeon or the radiologist to control and correct the segmentation of medical data sets.

1.3 Project Aims

The main objective of this project is to conduct a thorough research to explore, learn, understand and eventually be able to utilize Amira developer version 3.1 to perform segmentation and visualization on 3-dimensional Computed Tomography (CT) data sets.

1.4 Project Scope

The description presented in this project is intended for the users working in medical field such as doctors, surgeons, radiologists and physicians. Besides that, it can also assist CAD / CAM (Computer-Aided Design / Computer-Aided Manufacturing) student in segmenting and visualizing three-dimensional Computed Tomography images.

1.5 Project Limitations

The exploration done in Amira focuses on reading CT data, segmentation and visualization. Images from other modalities are not dealt with. The term *modalities* here refer to imaging techniques such as Magnetic Resonance Tomography (MRT), Positron Emission Tomography (PET), and Ultrasound. The usage of Amira is not limited to segmentation and visualization, however, description on other features such as volume image reconstruction, restoration, registration and texture mapping are not discussed here.

1.6 Project Development Schedule

Time management is a critical element in project development. It serves as a vehicle in determining how deep the problem has been analyzed, how comprehensive should a solution be designed to overcome the problem, how complete a source code can be implemented and how reliable is the end system.

The project schedule describes the system development cycle for this project. It helps to systematically organize the project by partitioning it into various discrete parts that can be done within a period of time. This enables the project developer to organize and rephrases each step accordingly. Figure 1.2 below shows the work breakdown and time allocation for each job that has been done.

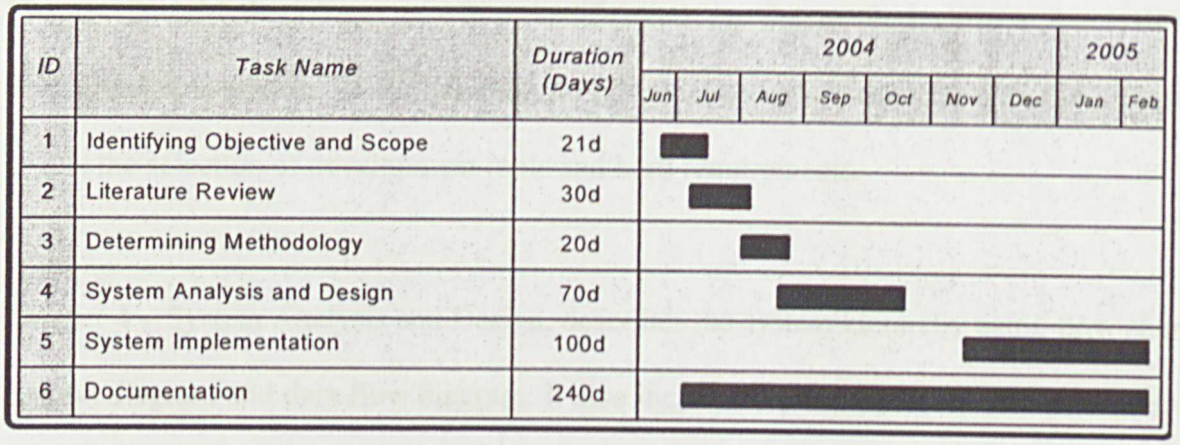


Figure 1.2: Project Development Schedule

As shown in Figure 1.2, documentation phase is an ongoing process, which is employed throughout the project development. All six phases have been successfully implemented.

1.7 Report Layout

This document contains several chapters that elaborate on various aspects from the start to the end of the system. These chapters and its content are arranged as following:

Chapter 1 – Introduction, gives an overview of the project including its problem definition, objective, scope, limitation and project development schedule.

Chapter 2 – Literature Review, discusses on the research done on segmentation of three dimensional computer tomography images. These include the segmentation techniques as well as the tools and software systems that are available for use.

Chapter 3 – Methodology, provides justification on the method chosen for system development as well as functional and non-functional requirements. Besides that, it also covers the selection of development tools and hard requirements.

Chapter 4 – System Analysis and Design, describes the system hierarchy using flow chart, context diagram and data flow diagram. It also includes a prototype of user interface and a statement of expected outcome.

Chapter 5 – System Implementation, explains how to get started to interacting with the working environment and interfaces of Amira. It demonstrates a step by step guide on how segmentation is done and ways to visualize the results.

Chapter 6 – Discussion and Conclusion, elaborate on the difficulties faced and ways to overcome them. System strength, limitation and enhancements are also included. Eventually, a conclusion is made to sum up the whole project development.

1.8 Summary

This chapter gives an overview of the project including the problem definition, aims, scope and limitations. It also dwells into the importance and feasibility of the project. Finally, project development schedule and report layout are provided for further depiction of the project development.

CHAPTER 2 – LITERATURE REVIEW

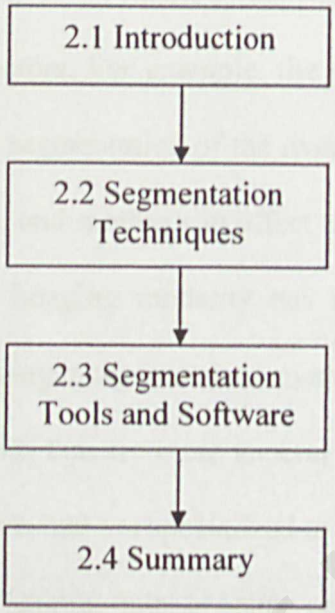


Figure 2.1: Overview of Chapter 2

2.1 Introduction

This chapter will discussed the previous works and researches done on medical images segmentation. It consists of the various techniques adopted in two-dimensional and three-dimensional image segmentation such as morphological technique, thresholding approaches, atlas-guided approaches and etc. Besides that, existing segmentation tools and software system will be covered as well. This includes the algorithms used in performing segmentation.

2.2 Segmentation Techniques

There are many types of segmentation methods depending on the specific application, imaging modality, and other factors. For example, the segmentation of brain tissue has different requirements from the segmentation of the liver. General imaging artifacts such as noise, partial volume effects, and motion can affect the performance of segmentation algorithms. Furthermore, each imaging modality has its own manner with which to contend. There is currently no single segmentation method that gives acceptable results for every medical image. Methods that are more general do exist and can be applied to a variety of data. However, methods that are specialized to particular applications can often achieve better performance by taking into account prior knowledge. Therefore it is extremely difficult to select of an appropriate approach to a segmentation problem. [4]

One of the key word in image segmentation is dimensionality. Dimensionality refers to whether a segmentation method operates in a 2D image domain or a 3D image domain. Methods that rely solely on image intensities are independent of the image domain. However, certain methods such as deformable models, Markov random fields, and region growing, incorporate spatial information and may therefore operate differently depending on the dimensionality of the image. Generally, 2D methods are applied to 2D images and 3D methods are applied to 3D images. However, in some cases, 2D methods are applied sequentially to the slices of a 3D image. This may occur because of practical reasons such as ease of implementation, lower computational complexity, and reduced memory requirements. In addition, certain structures are more easily defined along 2D slices. [4]

Segmentation techniques can be broadly classified into three classes:

1. Stochastic Techniques
    a. Thresholding Approaches
    b. Classification Techniques
    c. Clustering Algorithm
    d. Markov Random Fields

2. Structural Techniques
    a. 3D Edge-Detection Technique
    b. Morphological Technique
    c. Graph-Searching Algorithms
    d. Deformable Models
    e. Isosurfaces and Level Sets

3. Hybrid Techniques
    a. Region Growing
    b. Split and Merge
    c. Artificial Neural Networks
    d. Atlas-Guided Approaches
    e. LEGION Based

4. Other Approaches
    a. Model-Fitting Method
    b. The watershed algorithm

Among the methods discussed in this section, thresholding, classifier, clustering, and Markov random field approaches can be considered pixel classification methods. Note that in 2D discrete images, the location of each measurement is called a *pixel* and in 3D images, it is called a *voxel*. However, for simplicity, the two terms are used interchangeably. [4]

2.2.1 Stochastic Techniques

This technique performs segmentation by statistical analysis only. It does not take into consideration any structural information, thus it can be considered as 2D level segmentation. Examples of Stochastic Techniques are Thresholding Approaches, Classification Techniques, Clustering Algorithms and Markov Random Fields.

Thresholding Approaches

Image segmentation by thresholding is a technique for locating regions in an image that have the same property. There are two steps in the process. First, the image elements must be classified according to some criteria, usually the pixel value. For example, the user can define a maximum pixel value (a threshold) or a user-defined range of pixel

values (a threshold window) to separate the object(s) of interest from the remainder (background) of the image. Thresholding assumes that any pixel meeting the criteria for selection, no matter where it lies, is part of the object. Since there is no general theory of image segmentation, the process can be illustrated with two algorithms: a generic algorithm and Farrell's algorithm. [11]

Assume a 2D scene composed of pixels in which there is one object. The values assigned to the pixels lying within the bounds of the object fall within a continuous range of numbers, and no pixel outside the object has a value within this range. Scene size and the screen resolution are identical. The goal of the procedure is to form a binary scene. The procedure forms a binary scene by assigning all the pixels within the object a value of '1' and all other pixels a value of 'O'. One way to accomplish this task is as follows. First, the user indicates a range of pixel values that encompasses the range of pixel values for the object. [11]

Second, starting at the upper-left-hand corner of the scene, determine the binary scene values for screen pixels as follows: If a scene pixel has a value within the range, assign the corresponding screen pixel a value of '1'; otherwise assign the screen pixel a value of 'O'. Perform this operation for all the scene pixels. When the procedure terminates, display the object on the screen. [11]

Farrell's technique is also straightforward. It differentiates the object of interest from the remainder of the volume by assigning a unique color to the range of pixel values that

specifies an object. This technique assumes that the object corresponds to a continuous range of values (i.e., there are no discontinuities in the window) and that no other objects are incorrectly assigned the same color. [11]

The results are very dependent with the thresholds value used, which is the main drawback of this technique. Any change in the threshold values can give a different segmented region. The thresholds are usually generated interactively by using visual feedback. Some automatic methods do exist with varying degree of success to automate the process of finding correct thresholds. Another drawback is that this technique is very sensitive to noise and intensity in-homogeneities. [11]

Classification Techniques

This is a pattern recognition technique that tries to partition a feature space derived from the volume using data with known labels. A feature space is the range of an N-dimensional feature vector made from features at each pixel. Mathematically, a feature space can be the range space of any function of the volume. The features could include the pixel intensity, the gradient at the pixel, the distance of the pixel from the volume boundary and so on. [13]

As they require training data that are pre-segmented (either manually or by other method), classifiers belong to the supervised category. The pre-segmented data is then used as reference to carry out automatic segmentation on new data. [13]

The nearest-neighbor classifier is the simplest form of classifier, where each pixel or voxel is classified in the same class as the training data with the closest intensity. The k-nearest-neighbor (kNN) classifier is the generalization of this approach, where the pixel is classified according to the majority of the k closest training data. Another example of a similar classifier is the Parzen window, where the classification is made according to the majority vote within a predefined window of the feature space centered at the unlabeled pixel (mapped to feature space). Both of these classifiers are non-parametric since they do not make any assumption about the statistical structure of the data. [13]

Standard classifiers require that the structure to be segmented have distinct quantifiable features. Because training data can be labeled, classifiers can transfer these labels to new data as long as the feature space sufficiently distinguishes each label. Being non-iterative, they are relatively efficient and unlike thresholding, they can be applied to multi-channel volumes. A disadvantage of classifiers is that they generally do not perform any spatial modeling. This limitation has been addressed by incorporated neighborhood and geometric information. Another disadvantage is that manual interaction is required for obtaining training data. Training sets can be acquired from each volume that requires segmentation, but this can be difficult and time consuming. On the other hand, using the same training set for a large number of scans can lead to biased results which do not take into account anatomical and physiological variability among different subjects. [4]

Clustering Algorithms

This technique uses the characteristics of each pixel and its immediate neighborhood to do clustering. Clustering can be loosely defined as the process of grouping objects into groups, whose members show similar properties. In this case, these "objects" are the data pixels and the "groups" are the segmented regions. Similar "properties" could be any property of the data pixel, such as the density, gradient, color (for a color dataset) etc. [13]

This method is similar to the classifier methods mentioned above except that it does not use any training data. Thus, it comes under the unsupervised class of algorithms for segmentation. These algorithms overcome the need for a training data by iterating between segmenting the volume and characterizing the properties of each class. It can be said that clustering-based algorithms train themselves using the available data. The various clustering algorithms available today can be grouped into two broad categories:

1. Hierarchical methods: These methods include those techniques where the input data is not partitioned into clusters in a single step. A series of successive fusions of data are performed until each cluster of size greater than one is composed of smaller clusters.

2. Non-Hierarchical methods: In these methods, the desired number of clusters is known or assumed at the beginning of the clustering process. Each data pixel is assigned to exactly one cluster at the end result in this algorithm.

In the case of classification mentioned above, pixel properties such as intensity, gradient, neighborhood information etc. are used to form an N-dimensional feature vector for each pixel. Each class of the region is assumed to form a distinct cluster in the N-dimensional feature space. A suitable clustering algorithm, (such as K-means clustering, leader clustering, spatial clustering, etc.) is then applied to each pixel in the feature space. The resultant clusters in the feature space are then mapped to spatial domains to give the desired regions. [13]

Markov Random Field

Markov random field (MRF) modeling itself is not a segmentation method but a statistical model which can be used within segmentation methods. MRF models spatial interactions between neighboring or nearby pixels. These local correlations provide a mechanism for modeling a variety of image properties. Most pixels are thought to belong to the same class as their neighboring pixels. In physical terms, this implies that any anatomical structure that consists of only one pixel has a very low probability of occurring under a MRF assumption. MRFs are often integrated into clustering segmentation algorithms such as the K-means algorithm under a Bayesian prior model. The segmentation is then obtained by maximizing the *a posteriori* probability of the segmentation given the image data using iterative methods such as iterated conditional modes or simulated annealing. [4]

The selection of proper parameters controlling the strength of spatial interactions is a difficulty associated with MRF models. A setting too high can result in an excessively smooth segmentation but a loss of important structural details. In addition, MRF methods usually require computationally intensive algorithms. Despite its disadvantages, MRFs are widely used not only to model segmentation classes, but also to model intensity in-homogeneities that can occur in MR images and texture properties. [4]

2.2.2 Structural Techniques

The segmentation methods that fall under this category try to find structural properties of the region to be segmented. Structural properties such as intersecting surfaces (edges in 2D) are detected in the volume and then combined to segment the region. In some algorithms, structure information is saved and later retrieved to perform segmentation on similar datasets. Examples of structural techniques are 3D Edge-Detection Technique, Morphological Technique, Graph-Searching Algorithms, Deformable Models, Isosurfaces and Level Sets.

3D Edge-Detection Technique

This technique aims at detecting edges or surfaces in the volume to perform segmentation. Edges are formed at the intersection of two regions with different intensities. They are

one of the main cues for visual distinction of two regions. This technique works in two stages:

1. Local edges are detected by using some form of differentiation.
2. These local edges are grouped together to form boundary contours that separate the desired region voxels from other voxels.

A 3D surface detection algorithm that extends the classical Robert's operator into 3D space has been proposed. Researches later extended this algorithm to 4D. An optimal three dimensional edge detection operator, which was essentially a Sobel operator, was also developed. One advantage of edge detection techniques is that they work very well on datasets with good contrast between different regions. The edges are detected perfectly and can be verified visually. However, on the down side, these algorithms detect all the edges. Therefore, it is very difficult to find the correlation between the edges and the region-of-interest. In addition, these algorithms do not perform well on datasets with low contrast between regions; they are also susceptible to noise. This technique is usually not used on their own for segmentation, but combine with other segmentation algorithms to solve a particular segmentation problem. [13]

Morphological Techniques

Mathematical morphology uses set transformations for image analysis. It extracts the impact of a particular shape on images via the concept of structuring elements (SE). The

SE encodes the primitive shape information. The shape is described as a set of vectors referenced to a particular point, the center. During morphological operations, the center scans the whole image and the matching shape information is used to define the transformation. The transformed image is thus a function of the SE distribution in the whole image. Dilation and erosion are the most fundamental transforms in mathematical morphology, thus they are discussed here:

1. Dilation: performed by lying the SE on the image and sliding it across the image. The steps involved are:

    a. If the origin of the SE coincides with a '0' in the image, there is no change; move to the next pixel.
    b. If the origin of the SE coincides with a '1' in the image, perform the 'OR' logic operation on all pixels within the SE.

2. Erosion: similar to dilation but turns pixels to '0', not '1'. The steps involved are:

    a. If the origin of the SE coincides with a '0' in the image, there is no change; move to the next pixel.
    b. If the origin of the SE coincides with a '1' in the image, and any of the '1' pixels in the SE extend beyond the object, then change the '1' pixel in the image to a '0'. [1]

Morphological operations are generally simple to understand and easy to implement. At the same time, they are also difficult to control. For example, it is difficult to control the dilation operation unless the user gives the upper limit to the number of times it dilates. Thus, these algorithms generally require some external criteria to control them. These operations also have a risk of changing the morphology of the input datasets. [14]

Graph-Searching Algorithms

In this algorithm, edges and surfaces in a volume are represented as graphs and it tries to find the lowest-cost path between two nodes of the graph using a search algorithm such as A* and F*. These algorithms are very useful when the partitions between regions in the desired segmented volume are not well defined. The F* algorithm is used extensively in biomedical imaging and hence we will discuss it here. [13]

Generally, both the F* and the A* are quite similar. In A* algorithm, a minimum-cost path from the starting point (s) to the goal point (g) is iteratively constructed by extending the best partial path available at each iteration. This is done by selecting the point v that has the minimum cost path from s to g via v where the cost is the sum of lowest-cost paths found so far from s to v and the estimate of minimum cost paths from v to g. For a simple implementation the algorithm requires O (N^2) operations. Similarly, the F* algorithm finds the optimum path from s to g using a cost array C by iteratively updating a path array P. This array (P) is initialized to infinity except at s, which is set to C(s). Due

to its complexity, the mathematical derivation for this algorithm will not be included here. [13]

This method has an advantage that is it can perform well even if the partitions between regions are broken. However, it requires these surfaces to be represented as graphs, which could be complicated. Another disadvantage from volume visualization point of view is that this algorithm deals with surfaces. To get the voxel representation of these surfaces, another pass is needed to convert the surfaces to voxels. [13]

Deformable Models

Deformable models are physically motivated, model-based techniques for delineating region boundaries using closed parametric curves or surfaces that deform under the influence of internal and external forces. In order to delineate an object boundary in an image, first a closed curve or surface must be placed near the desired boundary and then allowed to undergo an iterative relaxation process. Internal forces are computed from within the curve or surface to keep it smooth throughout the deformation. External forces are usually derived from the image to drive the curve or surface towards the desired feature of interest. [4]

Physically based deformable models can be divided into three categories:

1. Energy minimizing snakes: It is the most popular form of deformable models. Snakes are planar deformable contours that are useful in several image analysis tasks. Using energy minimization formulation, the aim of this approach is to find a parametric model that minimizes the weighted sum of internal energy and potential energy. The internal energy specifies the tension or the smoothness of the surface of the model. The potential energy is defined over the volume domain and typically possesses local minima at the edges occurring at object boundaries. Minimizing the total energy yields internal and potential forces. As a result, these are attracted to image features such as lines and edges. [9]

2. Dynamic deformable models: Although it is natural to think of energy minimization as a static problem, a compelling approach to compute the local minima of functional is to construct a dynamical system that is governed by the functional and allow the system to evolve to equilibrium. Equilibrium is achieved when the internal and external forces balance and the contour comes to rest. This leads to dynamic deformable models that unify the description of shape and motion, making it possible to quantify not just static shape, but also shape evolving through time. [9]

3. Probabilistic deformable models: Deformable models can also be viewed as a model fitting process in a probabilistic framework. This allows the incorporation of prior model and sensor model characteristics in terms of probability distributions. The probabilistic framework also provides a measure of the

uncertainty of the estimated shape parameters after the model is fitted to the image data. [9]

The advantage of Deformable models is that they offer a logical and consistent mathematical description. They are also robust to noise and boundary gaps due to their incorporation of a smoothness constraint. Another outcome is that they offer sub-voxel accuracy for the boundary representation that may be important to some applications. A very important advantage of these models from the medical imaging point of view is that they are capable of accommodating the often-significant variability of biological structures over time and across different individuals. [4]

The main drawback is that they require manual interaction to place an initial model in the dataset. These algorithms also require the user to choose appropriate initial parameters. Various methods have been proposed to reduce sensitivity to initialization. Another disadvantage from the volume visualization point of view is that they only work on surfaces. Therefore, there is a lot of work being done to extend this idea to volumetric solid models. [4]

Isosurfaces and Level Sets

Isosurfaces are defined by connecting voxels with intensities equal to the iso-value in a 3D volume. Level sets, on the other hand, are moving fronts (curves). The idea behind this technique is to use isosurfaces as a modeling technology that can serve as an alternative to parameterized models. [7]

Level-sets are numerical techniques designed to track the evolution of interfaces, which in this case would be the isosurface. Other numerical techniques try to follow moving boundaries by putting a collection of marker points on the evolving surface and then changing their position to correspond to the moving surface. In contrast, level-set methods exploit a strong link between moving surfaces and equations from computational fluid equations. Rather than follow the interface itself, the level set approach takes the original curve, and build it into a surface. [13]

The representation of level-set has a number of practical and theoretical advantages over conventional surface models, particularly in the context of deformation and segmentation. First, level-set models are topologically flexible, they can easily represent complicated surface shapes that can form holes, split to form multiple objects, or merge with other objects to form a single structure. Second, these models can incorporate many (millions) degrees of freedom, and therefore can accommodate complex shapes. Thus, there is no need to re-parameterize the model as it undergoes significant changes in shape. [13]

2.2.3 Hybrid Approaches

In this section we will look at the segmentation algorithms which cannot be classified into the previous two categories. These algorithms use something from both the previous two types of segmentation algorithms. Examples of Hybrid Approaches are Region Growing, Split and Merge, Artificial Neural Network, Atlas-Guided Approaches and LEGION Based.

Region Growing

This technique aims to extract a connected region from a 3D volume based on some pre-defined connecting criterion. This criterion can be as simple as the voxel intensity or could be the output of any other segmentation algorithm. In the simplest form, region growing requires a seed point to start with. From the seed point, the algorithm grows until the connecting criterion is satisfied. [13]

Like thresholding, region growing is simple, but not often used for segmentation by itself. It is used as part of segmentation techniques for a particular approach. Region growing is often used as the primary method to understand a 3D data before more complex segmentation is being applied. [13]

The main drawback of this algorithm is that it requires a seed point for each region to be segmented, which means manual interaction is needed. Region growing can also be sensitive to noise and partial volume effect causing the extracted region to have holes or

disconnections. Some recent work has been reported which tries to overcome these problems. [4]

Split and Merge

Split and merge is quite similar to region growing discussed earlier. It requires the input data to be organized into a pyramidal grid structure of regions, with each region organized into groups of eight (for 3D). Any region can be split into eight sub-regions and the appropriate eight can be merged into a single larger region. As in region growing, the criteria for merging (growing for region-growing) could be anything. It could be as simple as voxel intensity or some condition checking based on the output of some previous segmentation stage. The algorithm can be written as follows:

1. Define a homogeneity test. This involve defining a homogeneity measure, which may incorporate brightness, color, texture, or other application-specific information, and determining a criterion the region must meet to past the homogeneity test.
2. Split the image into equally sized regions.
3. Calculate the homogeneity measure for each region.
4. If the homogeneity test is passed for a region, then a merge is attempted with its neighbor(s). If the criterion is not met, the region is split.
5. Continue this process until all regions pass the homogeneity test. [1]

The main advantage of this method over region growing is that no manual interaction is needed, as no seed points are required. However, it requires the input to be organized into a pyramidal grid structure, which could be undesirable for huge datasets. [13]

Artificial Neural Networks

Most of the conventional segmentation algorithms discussed are based on structural knowledge and often require considerable user expertise. The Artificial Neural Networks (ANN) based approaches tried to partially overcome these drawbacks. ANNs are massively parallel networks of processing elements or nodes that simulate biological learning. Each node is capable of performing elementary computation. Learning is achieved through the adaptation of weights assigned to the connections between nodes.

The main features of ANNs that the segmentation algorithms try to use are:

1. Learning from examples and generalizing that knowledge
2. Noise rejection
3. Fault tolerance
4. Optimum seeking behavior

Researches have presented three architectures for medical image segmentation based on ANNs. These architectures showed that ANNs can successfully exploit and integrate

different kinds of prior information contained in medical images. The experiments demonstrated robustness and sensitivity of the approach, but at the minimum expenses. [4]

Artificial Neural Networks are widely used in segmentation as a classifier, where the weights are determined using training data, and then used to segment new data. The network can also be used in an unsupervised fashion as a clustering method, as well as for deformable models. Since the ANNs are tightly interconnected, spatial information can be easily incorporated into its classification procedures. Despite ANNs are inherently parallel, their processing is usually simulated on a standard serial computer, thus reducing its potential computational advantage. [4]

Atlas-Guided Approaches

A standard atlas or template is used to perform segmentation in this approach. The atlas is generated by compiling information on the anatomy that requires segmentation. This atlas is then used to segment other images. The standard atlas-guided approach treats segmentation as a registration problem. It first finds a one-to-one transformation that maps a pre-segmented atlas image to the target image that requires segmentation. This process is known to as atlas warping, which can be performed using linear transformations. [4]

An advantage of atlas-guided approaches is that labels are transferred as well as the segmentation. The main shortcoming of this method is due to anatomical variability. To

overcome this, many researchers have tried to apply a sequence of linear and non-linear transformations. Even with this, accurate segmentation of complex structures is extremely difficult. Researches have introduced probabilistic atlases to model anatomical variability, but their method required additional time and interaction to accumulate data. Therefore, these approaches are best suited for segmenting structures that are stable over the population of study. [4]

LEGION Based

These segmentation methods are based on a biologically inspired oscillator network, called the Locally Excitatory Globally Inhibitory Oscillator Network (LEGION). It was proposed as a biologically plausible computational framework for image analysis. The network was based on theoretical and experimental considerations that point to oscillatory correlation as a representational scheme for the working of the brain. [13]

The oscillatory correlation theory assumes that the brain groups and segregates visual features on the basis of correlation between neural oscillations. It has been shown theoretically and later, by experiments that neural oscillations in the visual cortex are a possible mechanism by which the brain detects and binds features in a visual scene. [13]

LEGION is a network of relaxation oscillators, each constructed from an excitatory unit x and an inhibitory unit y. Unit x sends excitation to unit y which responds by sending inhibition back. When external input stimulus I is continuously applied to x, this feedback

loop produces oscillations. Neighboring oscillators are connected via mutual excitatory coupling, as well as the global inhibitor. [13]

LEGION network is computationally expensive to simulate since it requires numerically integrating a huge number of differential equations. Therefore, it is almost impossible to segment large volume datasets using this technique. To make it feasible, researchers proposed a simplified algorithm for efficiency purposes, which are important from the volume segmentation point of view. [13]

The advantages of this method are: requires less intervention compared to most of the structural techniques, initial parameter setting can be fully automated, and can achieve good noise tolerance. However, due to generality, domain-specific knowledge is not fully utilized compared to structural techniques. [13]

2.2.4 Other Approaches

Model-fitting is a segmentation method that typically fits a simple geometric shape such as an ellipse or parabola to the locations of extracted image features in an image. It is a technique specialized to the structure being segmented but is easily implemented and can provide good results when the model is appropriate. A more general approach is to fit spline curves or surfaces to the features. The main difficulty with model-fitting is that image features must first be extracted before the fitting can take place. [4]

The watershed algorithm uses concepts from mathematical morphology to partition images into homogeneous regions. This method can suffer from over-segmentation, which occurs when the image is segmented into an unnecessarily large number of regions. Thus, watershed algorithms in medical imaging are usually followed by a post-processing step to merge separate regions that belong to the same structure. [4]

2.3 Segmentation Tools and Software Systems

2.3.1 3D Slicer

The 3D Slicer is a software tool used for surgical planning, surgical navigation, segmentation and registration of medical imagery. It uniquely integrates several facets of image-guided medicine into a single environment. Some of the key features in 3D Slicer are:

1. Automatic registration: aligning data sets
2. Semi-automatic segmentation: extracting structures such as vessels and tumors from the data
3. Generation of 3D surface models: for viewing the segmented structures
4. 3D visualization, and quantitative analysis of various medical scans: measuring distances, angles, surface areas, and volumes

Other common segmentation tools:

Thresholding: Selects and outlines various structures based on a certain pixel value. It is very useful for brain, ventricles, and skin models of the head for example. By manipulating the sliding scales on the threshold screen, user may select just the structures desired. After browsing through the original data images, apply functions to the data to be segmented.

Change Island: This is a very good function to use right after thresholding. It will change an outlined area, an "island", to the color that have been selected. For example, if the images of the brain have been threshholded and a brain model wish to be created, it will be necessary to remove outlines from the skull bones. By clicking on the respective skull bone while using a black "output" for color, the outlines around the skull will be removed.

Remove Islands: This is an automated function that uses island size as a parameter for removing small areas that may have been thresholded that should not have been. For example, if user input a value of "10", all of the islands in a particular image less than 10 pixels will be removed. [6]

2.3.2 3D-Doctor

3D-DOCTOR is advanced 3D image processing, modeling, and measurement software for MRI, CT, PET, microscopy, and industrial 3D imaging applications. It is used for 3D image visualization, volume rendering and deconvolution applications by researchers, doctors and engineers. Image segmentation has been improved to generate object boundaries from image slices, and create 3D rendering in a few steps. The vector-based boundary and contour editor provides a flexible tool for image editing and handling. Some of the features include:

1. Import Image File Formats
2. Export 3D Formats
3. Fully Automatic and Interactive 3D Image Segmentation
4. Fast 3D Surface Modeling
5. Real Time 3D Volume Rendering
6. 3D Volume Image Reconstruction
7. 3D Image Restoration by Deconvolution

3D-DOCTOR provides three types of 3D image segmentation functions: Fully Automatic Segmentation, Interactive Segmentation, and Segment Object Using a Training Area.

1. Fully Automatic Segmentation: 3D-DOCTOR uses the 3D image texture to separate objects and trace their boundaries accurately. The boundary data can be edited easily using the vector based Boundary Editor functions and exported to many graphics formats. Most 3D images can be segmented in just seconds or

minutes using this method. The extracted object boundaries are then used by both the 3D surface rendering and direct volume rendering functions.

2. Semi-automatic Interactive Segmentation: For each image plane, 3D-DOCTOR first estimates the optimal threshold using the image histogram derived from the current plane. The user can interactively adjust the threshold to make it perfect for the plane and then let 3D-DOCTOR segment the plane and extract all the boundaries. Once the image plane is segmented, simply switch to the next plane, and do the segmentation again.

3. Segment Object Using a Training Area: This method is best when extracting boundaries from a 3D object which has unique image attributes. It allows a user drawn image region to be used as a training area to segment the 3D image, based on the features generated from the area. This function is often used to extract certain objects inside a larger object, such as bones and blood vessels within the head in an MRI or CT image. [15]

2.3.3 3DVIEWNIX

3DVIEWNIX is a transportable, very inexpensive software system capable for visualizing, manipulating, and analyzing multidimensional, multi-modality image information. It can handle object information from multiple modalities and longitudinal

acquisitions. Multitudes of visualization, manipulation, and analysis methods such as below are incorporated.

1. Preprocessing
    a. Interpolation: To create isotropically sampled data of lower or higher resolution than input. Interpolation can be done in N-dimensions, both grey-level and shape-based methods.
    b. Thresholding: Multiple intervals can be specified; iso-surface can be generated at any resolution.
    c. Segmentation: 2-feature cluster partitioning or quick gesture-controlled (user-guided) boundary segmentation.
    d. Masking: To assists segmentation, quick operation using 'paint brushes'.
    e. Classification: 1-feature multiple material classification for opacity assignment or 2-feature multiple material classification for opacity assignment.

2. Visualization – Surface or volume rendering

3. Manipulation – Complex surgical procedures can be simulated. A variety of complex operations including cut away, reflect, separate, move, surface marking, measure and animation.

4. Analysis

   a. Measurement: Provides variety of image intensity-based measurements such as density profile, time density curves, region-of-interest statistics and their variation with time.

   b. Registration: Based on matching homologous features - points, curves, and entire surfaces. It is for merging information from multiple modalities, motion description and analysis. [16]

2.3.4 Mvox

Mvox is a software tool for visualization, segmentation and manipulation of a wide range of 2-4D grey level and colour images, and 3D surface graphics. The principal idea behind the software has been to provide a flexible tool that is able to handle all the kinds of data that are typically used in a research environment for medical imaging and visualization. At the same time the software is easy to use and have a consistent interface providing locally only the functions relevant to the context.

Mvox can read and write the following image formats: ANALYZE, TIFF, SGI, TGA, BMP, Inrimage, HIPS, BRIMG. For 3D surface graphics renderings it supports the 3D formats: Import: CyberWare, OFF, Flex3D - Export: VRML, DXF, Inventor, Flex3D.

Segmentation can be performed manually on both RGB colour and grey level images using the drawing and contour definition facilities, thus allowing manual segmentation of

3D CT images or 2D colour images of histological slices. For semi-automatic segmentation, Mvox provides interactive/automatic thresholding and statistical classification using discriminator analysis. Segmented images can be visualized along with the original images using either 3D iso-surfacing, Volume Rendering or the built-in fast display of orthogonal slices. In addition, stacked contours can be combined into 3D structures.

Additional features include Histogram, Histogram stretch, Statistics, Enlargement, Colourmaps, Overview window, Advanced command line interface, User defined functions and etc. [10]

2.3.5 CTMRedit

CTMRedit is a GUI tool for viewing CT and MRI images in three orthogonal planes, for manually or automatically segmenting regions of interest (ROI), and for interpolating between outline contours to create a 3D outline surface. Features of CTMRedit include display and I/O of 8-bit and 16-bit image formats, easy navigation of a large image database, and easy-to-use ROI editing tools. CTMRedit is written in the Matlab programming language, so the code is platform-independent, and easy to extend or re-use. It is available for free, and new functionality can be added by anybody familiar with Matlab.

When loading a "main image", CTMRedit also looks for images of the same subject in orthogonal image planes, which can be displayed next to the main image as "locator images." As a result, if the user needs to segment some area from an axial image, he/she can use information from both the coronal and sagittal images to help identify and eventually segment the axial image. A "locator line" is drawn on each locator image to mark the location of the main image plane, and "zoom lines" show the edges of the main image window. Both of these lines are calculated using an auxiliary text file (the ".cor" file) in which the user has specified positions of 4 corners of the image plane in 3D Cartesian coordinates.

Tools for editing ROI outlines are also user-friendly. Tools include the "add point", "add line" and "delete region" tools, and a "magic wand", which uses a seeded region growing algorithm to automatically outline a dark region selected by the user. To avoid a "pixelly-looking" display, images in CTMRedit are upsampled and interpolated to the resolution of the user's monitor. With an upsampled main image, it is possible for users to draw an ROI outline with errors that are smaller than the size of an image pixel. [8]

2.3.6 ARIES

ARIES (Advanced Radiological Image Enhancement and Segmentation) is an interactive, semi-automated, portable segmentation software package designed to provide the radiologist with a number of image processing tools that facilitate the measurement and

comprehension of clinically important variables that may be determined from a set of tomographic medical images.

Image segmentation may be performed in one of three ways:

1. Manually, in which the user specifically selects pixels to be included in the region of interest.
2. Automatically, in which a computer selects pixels based on a previously determined set of rules.
3. Semi-automatically, in which both manual and automatic methods are used in combination.

Manual segmentation methods include pixel selection, geometrical boundary selection, and tracing. Given normal image resolutions of 64 x 64 or greater, selection of individual pixels is clearly impractical and rarely used. The selection of groups of pixels by boundary specification is much less time consuming, especially if the boundary can be approximated by simple polygons or ellipses. Though it is more time-intensive, manual tracing is both more accurate and flexible than simple geometrical approximation.

Fully automatic segmentation methods are usually impractical due to image complexity and the variety of image types and interpretations. Assuming a priori information regarding the location and type of region to be segmented, examples of fully automatic methods could include thresholding, boundary tracking, mathematical morphology and region growing. However, in most cases, this a priori information is not readily available

to the computer 4 prior to segmentation without user interaction. In addition, low contrast between structures generally causes even most "robust" automatic algorithms to fail.

Semi-automatic segmentation methods combine the benefits of both manual and automatic segmentation techniques. By supplying initial information about the region of interest, the user may guide an otherwise automatic segmentation procedure. By adjusting this information, the user may use the results of the segmentation procedure as increasingly accurate approximations to the actual region. Finally, any remaining errors introduced by automatic segmentation methods may be corrected by manual editing. [5]

2.3.7 NIHmagic

NIHmagic is a system for the visualization of three-dimensional biomedical images incorporating a sophisticated user interface, real-time object manipulation, manual registration, interactive segmentation, quantization of segmented objects, and volume rendering with texture mapping with gray scale or color lookup tables. This comprehensive tool will enhance medical treatment planning using biomedical imaging modalities such as MRI, CT, PET, Ultrasound, as well as electron microscopy.

NIHmagic is a visualization tool for research and clinical use. Images are reconstructed into a three-dimensional volume by volume rendering, a display technique that employs three-dimensional texture mapping to provide a translucent appearance to the object. A stack of slices is rendered into a volume by an opacity mapping function, where the

opacity is determined by the intensity of the voxel and its distance from the viewer. Functions included in NIHmagic are:

1. Volume rendering: Volume rendering is a direct representation of a three-dimensional data set. Unlike surface rendering, this technique does not require a priori knowledge of the surface for intermediate generation of geometrical representation. A geometric approximation of a surface within a volume set is formed from a cross connection of data points of equal value or density specified by a threshold value.

2. Texture mapping: It is used as the fundamental graphics primitive for volume rendering. The basic principle behind three-dimensional texture mapping is to apply an image (the texture) onto a three-dimensional polygon in a scene. The volumetric data is copied into the three dimensional texture image, also called texture space, and then displayed back to front, with sufficiently small intervals, as a stack of parallel slice planes.
3. Registration: Patient motion is a frequently encountered problem in medical imaging. Perhaps the most problematic case is cardiac motion during cardiovascular MRI. It is not yet possible to eliminate cardiac motion, maintain high spatial resolution, and achieve high image quality in three-dimensional acquisitions. Using the NIHmagic, a set of interactive tools for object manipulation, registration for cardiovascular MRI is accomplished by manually aligning anatomical features in orthogonal cross sections of the heart volume.

4. Segmentation: NIHmagic features an interactive tool to segment a volumetric object by manually tracing the object's boundary. The system enables the user to traverse through the volume and trace an object on an arbitrary cross section through the data volume. The cross section is represented by the 'SliceView', which is always perpendicular to the viewing direction. The traced boundary of the object is displayed as a contour within the given slice. [12]

2.3.8 Amira

Amira is a 3D visualization and modeling system. It allows user to visualize scientific data sets from various application areas, e.g. medicine, biology, chemistry, physics, or engineering. 3D objects can be represented as grids suitable for numerical simulations, notably as triangular surface and volumetric tetrahedral grids. Amira provides methods to generate such grids from voxel data representing an image volume, and it includes a general-purpose interactive 3D viewer.

Amira is a modular and object-oriented software system. Its basic system components are modules and data objects. Modules are used to visualize data objects or to perform some computational operations on them. The components are represented by little icons in the object pool. Icons are connected by lines indicating processing dependencies between the components, i.e., which modules are to be applied to which data objects. Data objects of specific types are created automatically from file input data when reading such or as

output of module computations, modules matching an existing data object are created as instances of particular module types via a context-sensitive popup menu. Networks can be created with a minimal amount of user interaction. Parameters of data objects and modules can be modified in Amira's interaction area.

The biggest part of the screen is occupied by a 3D graphics window. Additional 3D views can be created if necessary. Amira is based on the latest release of the TGS (Template Graphics Software) Open Inventor graphics toolkit. In addition, several modules apply direct OpenGL rendering to achieve special rendering effects or to maximize performance. In total, there are more than 120 data object and module types. They allow the system to be used for a broad range of applications. User-defined extensions are facilitated by the Amira developer version.

Amira provides a large number of module types allowing user to visualize various kinds of scientific data as well as to create polygonal models from 3D images. All visualization techniques can be arbitrarily combined to produce a single scene. Moreover, multiple data sets can be visualized simultaneously, either in several viewer windows or in a common one. A built-in transformation editor makes it easy to register data sets with respect to each other or to deal with different coordinate systems. Some of the key features in Amira are:

1. Direct Volume Rendering: Light emission and light absorption parameters are assigned to each point of the volume. Simulating the transmission of light through

the volume makes it possible to display user's data from any view direction without constructing intermediate polygonal models.

2. Isosurfaces: Isosurfaces are most commonly used for analyzing arbitrary scalar fields sampled on a discrete grid. Applied to 3D images, the method provides a very quick, yet sometimes sufficient method for reconstructing polygonal surface models.

3. Segmentation: Amira also provides a component for 3D image segmentation with several special-purpose features. This component is called image segmentation editor. It offers a large set of segmentation tools, ranging from purely manual to fully automatic. Among others, the following tools are provided: brush (painting), lasso (contouring), magic wand (region growing), thresholding, intelligent scissors, contour fitting (snakes), contour interpolation and extrapolation, various filters including smoothing, cleaning, and connected component analysis. Although the display is slice-oriented, many tools can be applied in both 2D and 3D. Since the editor does not store contours surrounding regions but region labels, a unique and well-defined classification is guaranteed.

4. Surface Reconstruction: Once the interesting features in a 3D image volume have been segmented, Amira is able to create a corresponding polygonal surface model. The surface may have non-manifold topology if there are locations where three or more regions join.

5. Surface Simplification: Surface simplification is another prominent feature of Amira. It can be used to reduce the number of triangles in an arbitrary surface model according to a user-defined value.

6. Generation of Tetrahedral Grids: Amira allows users not only to generate surface models from data but also to create true volumetric tetrahedral grids suitable for advanced 3D finite-element simulations. These grids are constructed using a flexible advancing-front algorithm.

Amira support various types of file formats such as:

1. ACR-NEMA – a predecessor of the DICOM format for medical images (.r)
2. DICOM – standard file format for medical images (.rw)
3. JPEG Image Format – 2D image format with lossy compression (.rw)
4. Open Inventor – standard file format for 3D models (.rw)
5. STL – simple format for triangular surfaces, no connectivity (.rw)
6. VRML – virtual reality markup language for 3D models (.rw)

2.4 Summary

As conclusion, there are many types of segmentation techniques available but there is no single one that will produce best performance for every medical image. Thus, it is

important to consider prior knowledge in order to determine methods that are specialized to particular applications that yield best performance.

On the other hand, numerous tools and software system exist that provide means to perform medical image segmentation. Each of them has their own strengths and features and utilizes different approaches and algorithms to achieve interactive segmentation.

CHAPTER 3

METHODOLOGY

CHAPTER 3 – METHODOLOGY

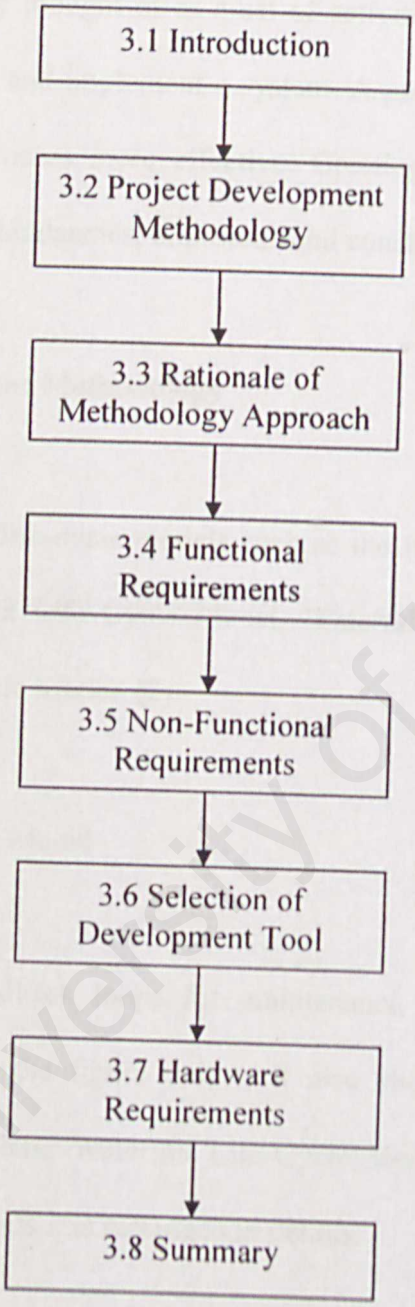


Figure 3.1: Overview of Chapter 3

3.1 Introduction

Methodology is classically thought of as a set of activities that analysts, designers and users carry out to develop and implement a system. A process model is built in order to make the development process more effective. Creating a process model can aid in finding inconsistencies, redundancies, omissions and constituent parts in the process.

3.2 Project Development Methodology

There are several types of life-cycle models such as the Iterative-and-Incremental Life-Cycle Model, Code-and-Fix Life Cycle Model, Waterfall Life-Cycle Model and The Rapid-Prototyping Life-Cycle Model. [2]

3.2.1 Waterfall Life-Cycle Model

This model shows the feedback loops for maintenance while the product is being developed. As reflected in the figure below, it also shows the feedback loops for postdelivery maintenance. Here, Waterfall Life-Cycle Model is chosen as the project development methodology, thus it is discussed in details.

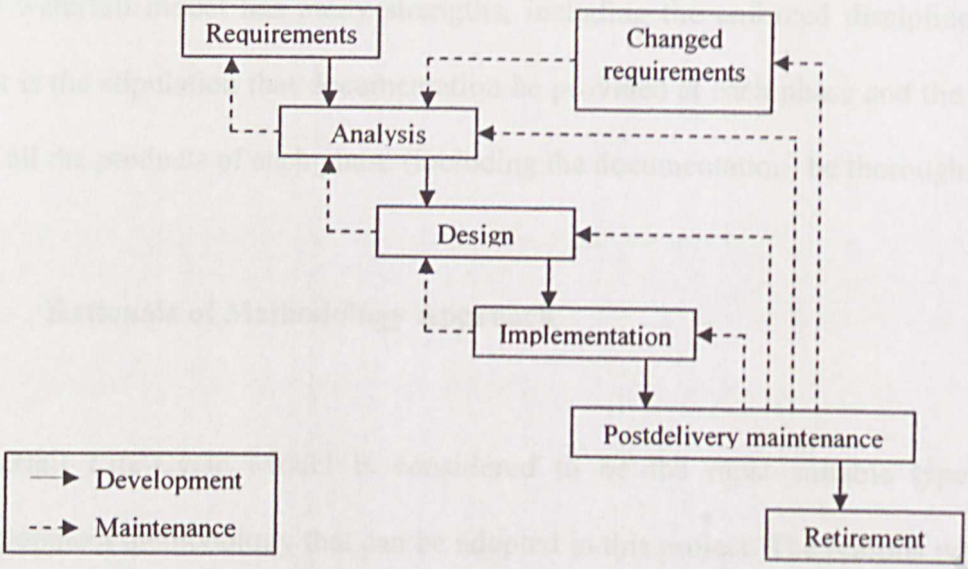


Figure 3.2: The Waterfall Life-Cycle Model

A critical point regarding the waterfall model is that no phase is complete until the documentation for that phase has been completed and the products of that phase have been approved. This carries over into modification; if the products of an earlier phase have to be changed as a consequence of following a feedback loop, that earlier phase is deemed to be complete only when the documentation for the phase has been modified and the modifications have been checked. [2]

Inherent in every phase of the waterfall model is testing. Testing is not a separate phase to be performed only after the product has been constructed, not is it to be performed only at the end of each phase. Instead, testing should proceed continuously throughout the software process. In particular, during maintenance, it is necessary to ensure not only that the modified version of the product still does what the pervious version did and still does it correctly, but that it also satisfies any new requirements imposed. [2]

The waterfall model has many strengths, including the enforced disciplined approach. That is the stipulation that documentation be provided at each phase and the requirement that all the products of each phase (including the documentation) be thoroughly checked.

3.3 Rationale of Methodology Approach

Waterfall Life-Cycle Model is considered to be the most suitable type of system development methodology that can be adopted in this project. The reasons why Waterfall Life-Cycle Model is chosen are:

- It present a very high level view of what goes on during development, the sequence of events that are expected to encounter is suggested. This is very useful in helping developers to lay out what they need to do.
- It enforces disciplined approach to develop a system as documents prepared after each stage will have to be checked.
- It supports good process visibility as each stage produces some kind of deliverable which may proved to be useful when the system evolves in the future.
- It enables maintenance to be carried out at each stage due to its interactive nature. Changes can be done during at any stages by returning to the previous stage.
- Many other models are actually just embellishments of the waterfall, incorporating feedback loops and extra activities.

System Requirement and Analysis

It contains the analysis on the system especially on the requirement analysis, which includes the functional requirements, non-functional requirements, hardware and software requirements. A requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose. It is important to understand what the users expect the system to do. Thus, we can say that our understanding of system intention and function starts with an examination of requirement. The kinds of item that are related to the requirements include the physical environment, interface, users and human factors, functionality, documentation, data, resources, security, quality assurance etc. [2]

System Design

The result of the requirement analysis is then translated into a solution: a design of the system. At this stage, the overall architecture of the system is established. The System Functionality Design that includes the System hierarchy and flow chart are drawn. Besides that, the system design also includes the graphical user interface. [2]

Implementation

This stage is where the setting up of the system environment takes place, which involve a lot of algorithms and new methods. Besides that, the system coding methodologies are

also included here. All programs will be coded using the selected programming language or application development tool.

Integration System Testing

All the units in the system are combined and tested as a whole system. The tasks at this stage include the verification and validations of the system to make sure the errors are at minimal level and the system meets the users' requirements. Enhancement will also be made to improve the quality of the system.

Operation and Maintenance

System development is complete when the system is operational. But a system's life does not end here. The continually evolving system needs to be maintained. Any work done to change the system after it is in operation is considered to be maintenance.

3.4 Functional Requirements

Functional requirements indicate a set of actions that the system is able to perform. It is often expressed in terms of inputs and outputs whereby given a specific input, the functional requirements stipulates what the output must be. The functional requirements involved in this project are discussed below.

3.4.1 Load Data

The system is able to load in data sets from supported file formats defined by users. It is able to load in single or multiple 2D slices at the same time. An error message will be displayed if the data being loaded is not a supported file format.

3.4.2 Display Images

This function is important for visualizing scalar data fields defined on uniform Cartesian grids, for example 2D and 3D image volumes. Such data are visualized by extracting an arbitrary axial, frontal, or sagittal slice out of the volume.

3.4.3 Label Voxel

The Label Voxel function provides a simple threshold segmentation algorithm. The method is also suitable for binary segmentation of other grey level images. Different regions can be extracted by applying suitable thresholding method.

3.4.4 Generate Surface

This computes a triangular approximation of the interfaces between different tissue types with either uniform or stacked coordinates. In other words, a surface is generated on the

segmentation result in order to view each part separately. The resulting surfaces can be non-manifold surfaces.

3.4.5 Save Data

This function enables the data / result to be saved either together or separately. The relationship between one data / module with another is denoted by a line in the object pool, they are consider a network and can be saved in batch form.

3.5 Non-Functional Requirements

A non-functional requirement is a description of the features, characteristic, and attributes of the system as well as any constraints that may limit the boundaries of the proposed solution. [2]

3.5.1 Maintainability

The system should be easy to maintain, flexible and effective. It is feasible to changes, modification and test in updating process to meet new requirements.

3.5.2 Reliability

A reliable system should maintain high consistency while functioning. The system should also operate in a user acceptable manner in the environment for which it was designed. It should not produce dangerous or costly failure when it is applied in a reasonable manner.

3.5.3 Portability

This system should be portable, which means it will work on various platform and computing environment.

3.5.4 Robustness

The system should be able to check for data validation before it continues for further processing to avoid unnecessary disaster. When a mistake is detected, it should prompt an error message to indicate the mistake before continuing.

3.5.5 User Friendly

The graphical user interface (GUI) should be user friendly and easy to understand by all level of users. It should ease the user to access to any part of the system and avoid using computer jargons for simplicity.

3.6 Selection of Development Tool

The development tool selected for this project is Amira version 3.1. It has a developer version (called AmiraDev), which allows us to write our own modules, data classes, editors, and I/O methods with any user interface desired. Amira also provides a component for 3D image segmentation with several special-purpose features. This component is called image segmentation editor. It offers a large set of segmentation tools, ranging from purely manual to fully automatic. Among others, the following tools are provided: brush (painting), lasso (contouring), magic wand (region growing), thresholding, intelligent scissors, contour fitting (snakes), contour interpolation and extrapolation, various filters including smoothing, cleaning, and connected component analysis. Although the display is slice-oriented, many tools can be applied in both 2D and 3D. Since the editor does not store contours surrounding regions but region labels, a unique and well-defined classification is guaranteed.

Amira 3.1 runs on Microsoft Windows 98/ME/NT4/2000/XP, HP-UX 11.00, SGI Irix 6.5.x, Sun Solaris 8 and 9, Linux (RedHat 8.0 with glibc-2.3.2.so or higher), and Linux IA64 (RedHat AW 2.1). Here, the selected operating environment is Microsoft Windows.

3.7 Hardware Requirements

The hardware requirements are quite minimal as all features are available even on low-end machine. However, for most applications it is highly recommended to have a

sufficiently large amount of main memory (for large amount of data sets) and to have a graphics card which supports both texturing and geometry processing in hardware. The recommended configurations are:

1. 500MHz Pentium 3 Processor
2. 128MB of Main Memory (preferably more)
3. 1GB Hard Drive
4. Other standard computer peripherals

3.8 Summary

This chapter described the methodology used in project development. Here, the waterfall life-cycle model is chosen and the rationale of this approach is discussed. Besides that, functional and non-functional requirements of the system are also included. Next, the software requirements such as the development tools, programming language and operating environment are covered. Last but not least, the hardware requirements such as machines and devices that are necessary for the system to run are stated.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

CHAPTER 4 – SYATEM ANALYSIS AND DESIGN

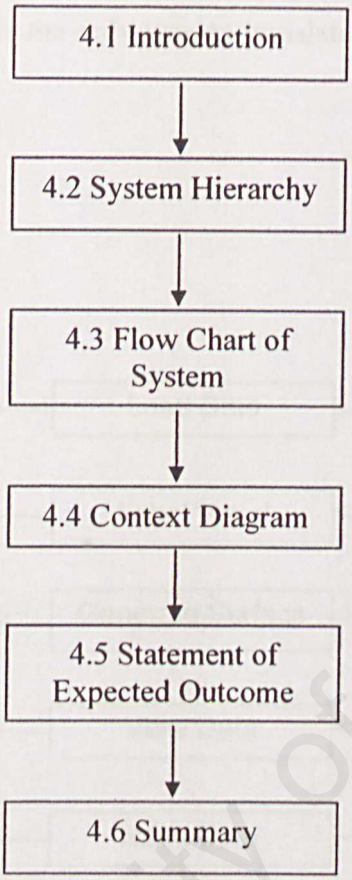


Figure 4.1: Overview of Chapter 4

4.1 Introduction

System analysis is an essential activity when new information systems are being built or the existing ones are changed. It is conducted for several purposes such as to identify the user's requirements, evaluate the system for feasibility and allocate functions to hardware, software, people and other system elements.

On the other hand, system design is a process through which requirements are translated into a model or representation of the software that can be assessed for quality before coding starts. System design is the only way to translate user's requirements accurately into a finished system.

4.2 System Hierarchy

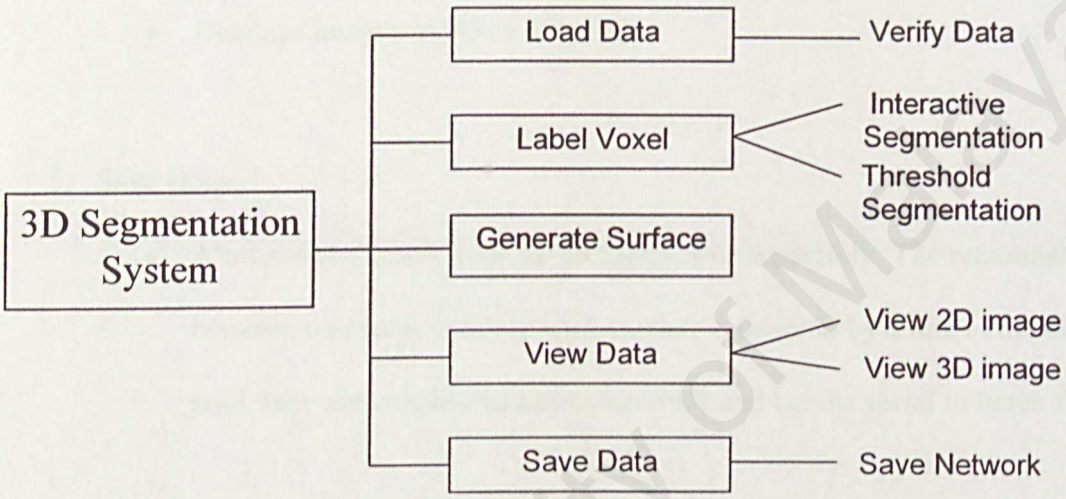


Figure 4.2: System Hierarchy

1. Load data
    - Load and verify the data for validity.

2. Label Voxel
    - Assign each voxel to a material using either interactive segmentation or threshold segmentation.

3. Generate Surface

   - This computes a triangular approximation of the interfaces between different tissue types with either uniform or stacked coordinates. In other words, a surface is generated on the segmentation result in order to view each part separately.

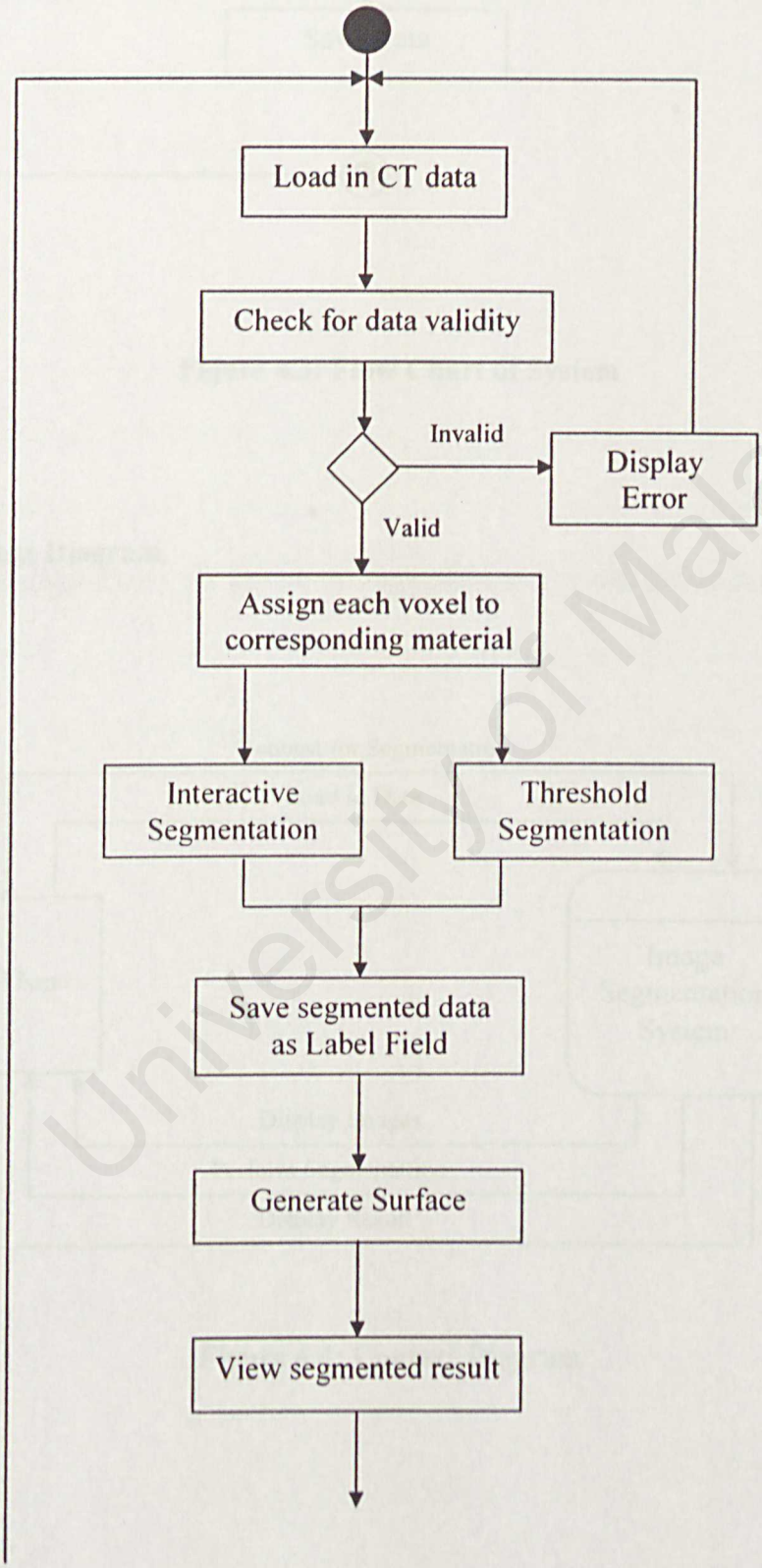
4. View Data

   - Displays images in 2D or 3D mode.

5. Save Data

   - Allows data / result to be saved together or separately. The relationship between one data / module with another is denoted by a line in the object pool, they are considered as one network and can be saved in batch form.

4.3 Flow Chart of System



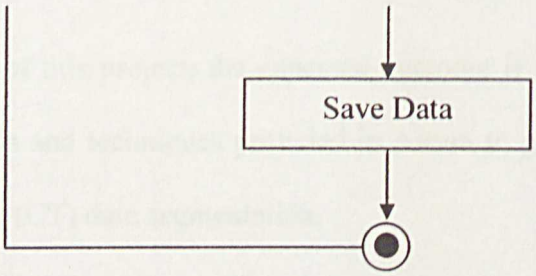


Figure 4.3: Flow Chart of System

4.4 Context Diagram

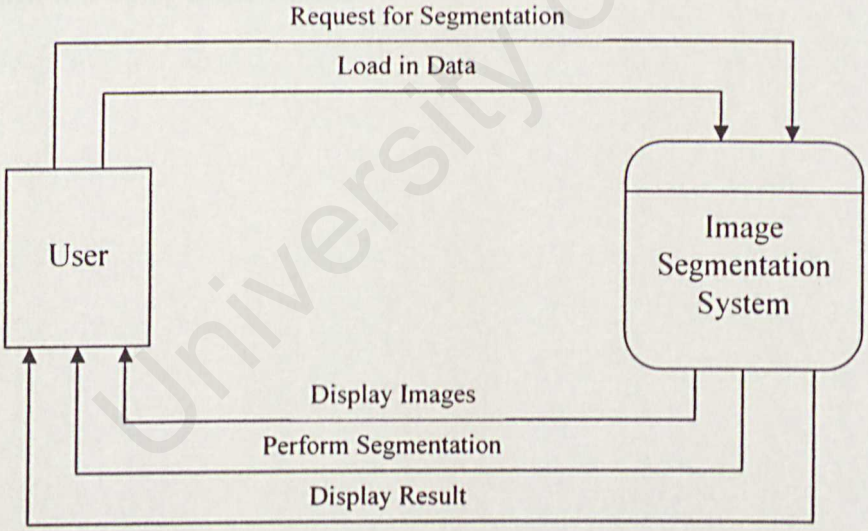


Figure 4.4: Context Diagram

4.5 Statement of Expected Outcome

At the end of this project, the expected outcome is to be able to understand and utilize various tools and techniques provided in Amira to perform three-dimensional computed tomography (CT) data segmentation.

4.6 Summary

This chapter described the system analysis and design phase in developing this project. It begins with an introduction, followed by the system hierarchy, flow chart of system and a context diagram. A prototype of the user interface design and a statement of expected outcome are included as well. These shall give an overview of how the system will look like and how it is being implemented.

CHAPTER 5

SYSTEM IMPLEMENTATION

CHAPTER 5 – SYSTEM IMPLEMENTATION

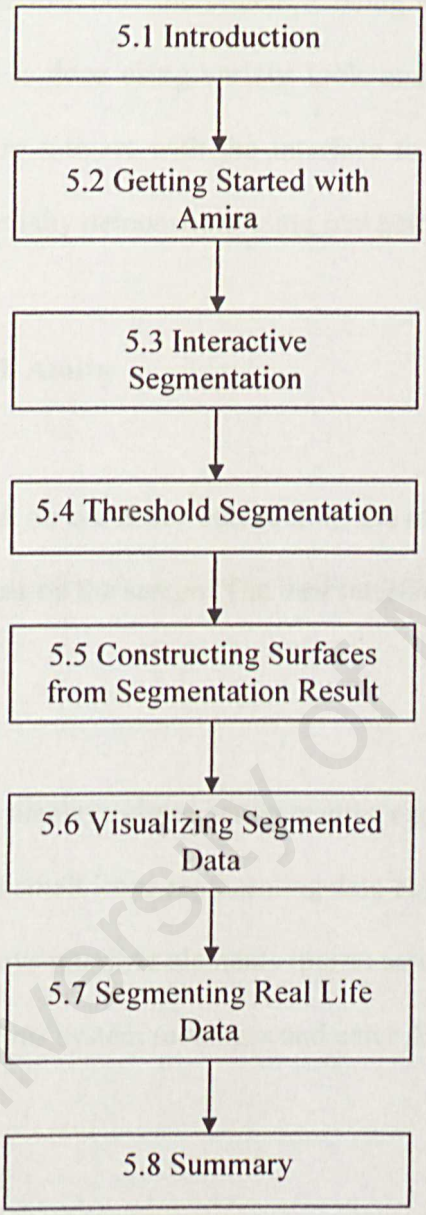


Figure 5.1: Overview of Chapter 5

5.1 Introduction

System implementation describes how the system is being accomplished. In this case, it explains how segmentation is done using various tools and techniques. Starting with a brief introduction on how to interact with the interface to detail information for each function in Amira, and eventually demonstrate using real patient's data.

5.2 Getting Started with Amira

To start running Amira, click on the ***Enter*** button from the main interface. Windows like those in Figure 5.2 will appear on the screen. The user interface is divided into four major parts (refer to Figure 5.2):

1. 3D viewer window – displays visualization results, e.g., slices or isosurfaces.
2. Object pool – contain small icons representing data objects and modules.
3. Working area – displays interface elements (ports) associated with Amira objects.
4. Console window – prints system messages and enter Amira commands.

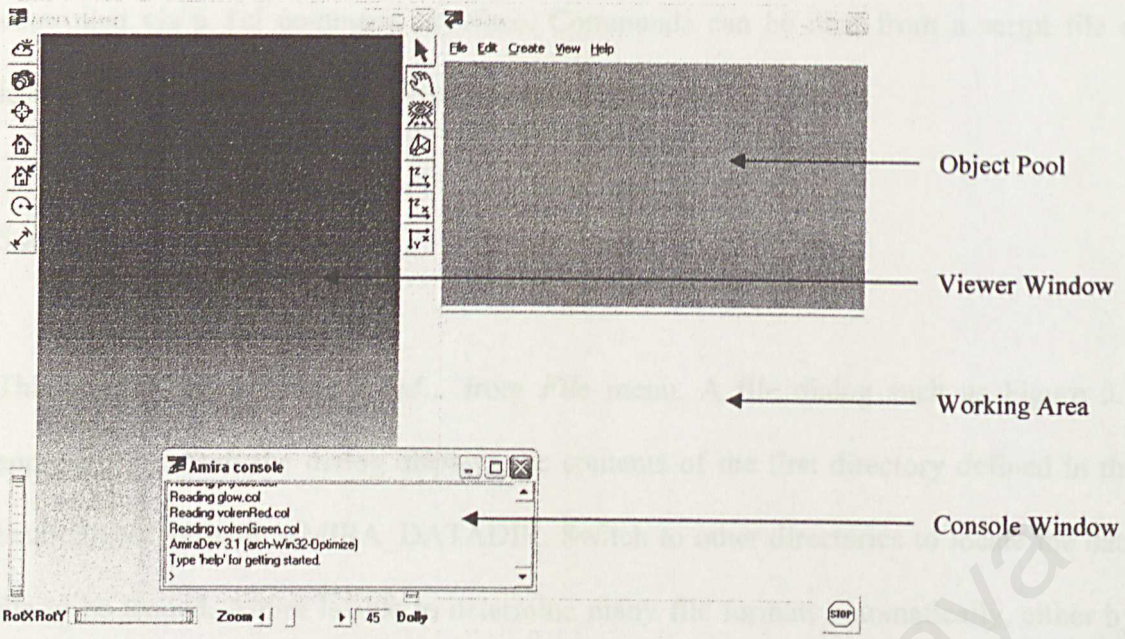


Figure 5.2: Amira Main Interface

Amira is a modular and object-oriented software system. Its basic system components are modules and data objects. Modules are used to visualize data objects or to perform some computational operations on them. The components are represented by little icons in the object pool. Icons are connected by lines indicating processing dependencies between the components, such as which modules are to be applied to which data objects. This type of connection is called networks which can be created easily. Parameters of data objects and modules can be modified in Amira's interaction area.

For some data objects such as surfaces or colormaps, there exist special-purpose interactive editors that allow the user to modify the objects. All components can be

controlled via a Tcl command interface. Commands can be read from a script file or issued manually in a separate console window.

5.2.1 Loading the Data Set

This is done by choosing *Load...* from *File* menu. A file dialog such as Figure 5.3 appears, by default the dialog displays the contents of the first directory defined in the environment variable AMIRA_DATADIR. Switch to other directories to locate the data file to be loaded. Amira is able to determine many file formats automatically, either by analyzing the file header or the file name suffix. The format of a particular file will be printed in the file dialog right beside the file name. Upon loading, for certain data which is not in Amira native file format (.am) such as DICOM, another file loader dialog such as Figure 5.4 appears displaying additional information such as patient's identity, type of modality, etc.

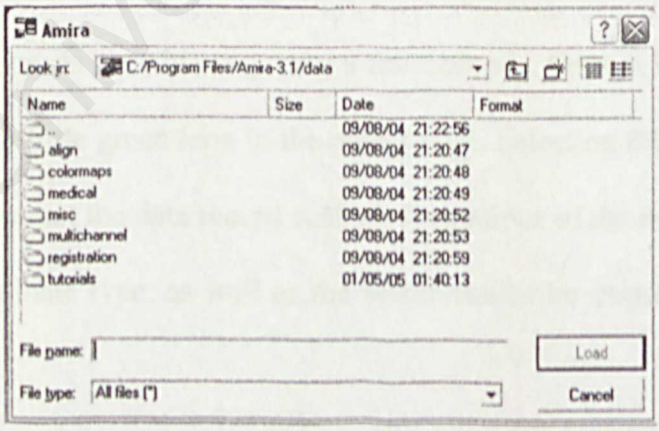


Figure 5.3: File Dialog

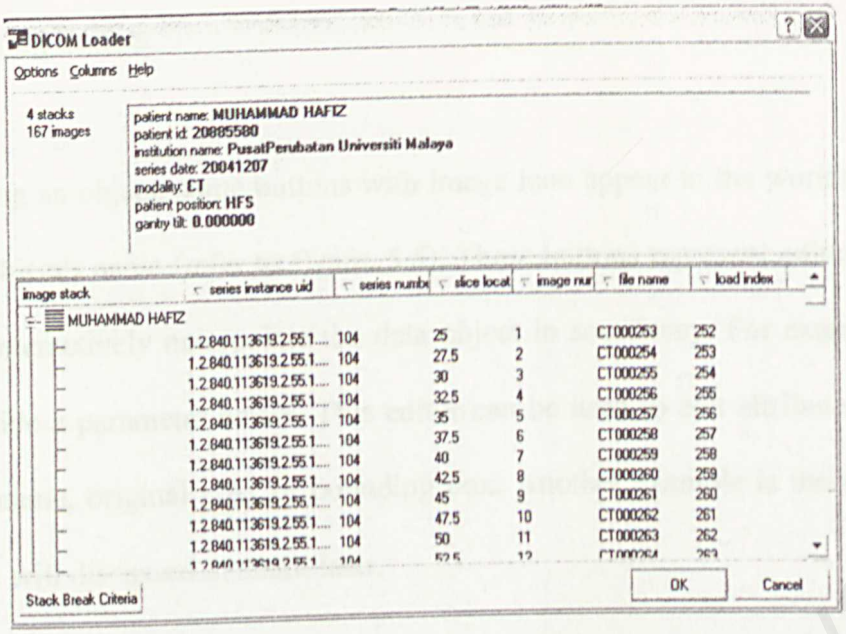


Figure 5.4: File Loader Dialog

Multiple 2D slices are used to create a 3D image data. The 2D images may be written in TIFF, BMP, JPEG, or any other supported image file format. In order to load such data, all 2D slices have to be selected simultaneously in the file browser. This can be done by clicking the first file and shift clicking the last one. The data will be loaded into the system. Depending on its size this may take a few seconds. Once it has been loaded, the data set appears as a little green icon in the object pool. Selecting this icon by clicking it causes information about the data record such as dimensions of the data set, the primitive data type, the coordinate type, as well as the voxel size to be displayed in the working area.

5.2.2 Invoking Editors

After selecting an object, some buttons with image icon appear in the working area, next to the data object's name (refer to Figure 5.5). These buttons represent editors which can be used to interactively manipulate the data object in some way. For example, all data objects provide a parameter editor. This editor can be used to edit attributes of the data set, e.g. filename, original size, or bounding box. Another example is the segmentation editor which will discussed in detail later.

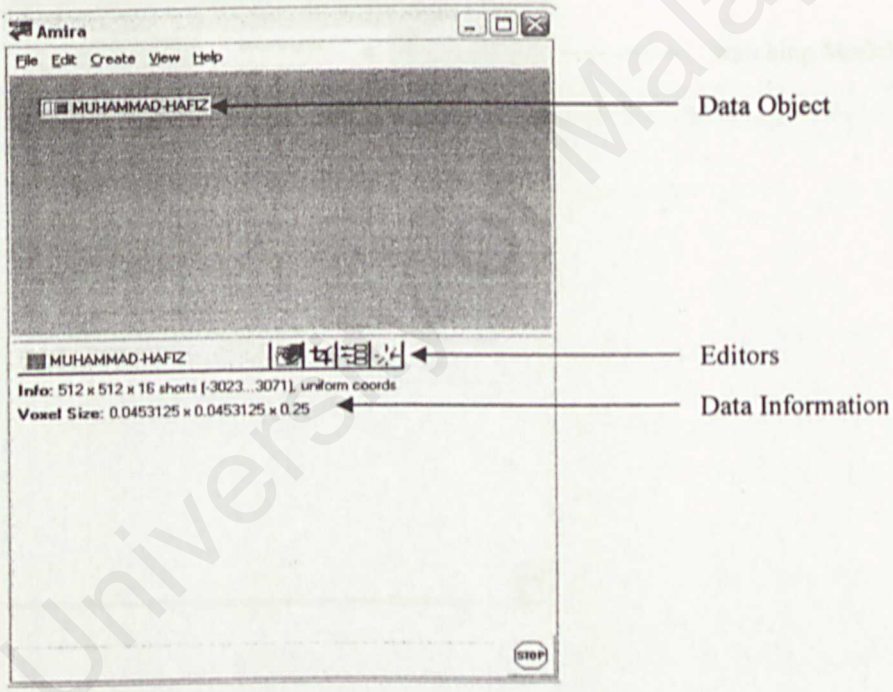


Figure 5.5: Data Objects and Editors

5.2.3 Connecting Visualization Modules

Data object can be visualized by attaching display modules to them. Each icon in the object pool provides a popup menu from which matching modules, e.g., modules that can operate on this specific kind of data, can be selected. To activate the popup menu, right click on the data icon and choose the appropriate module, e.g., *OrthoSlice*. A new module is created and is automatically connected to the data object (refer to Figure 5.6).

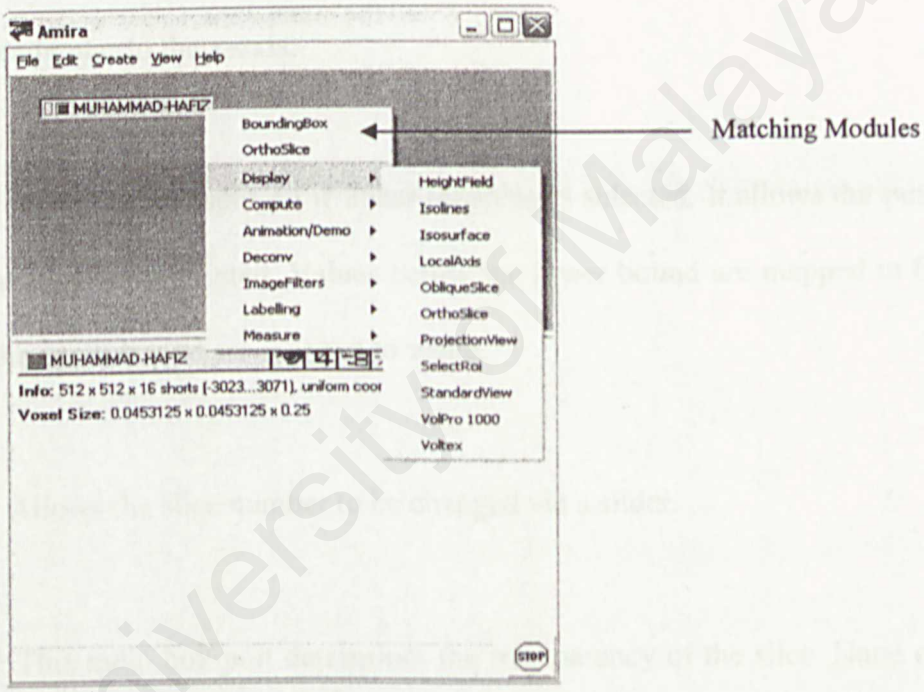


Figure 5.6: Connecting Modules

The *OrthoSlice* object is represented by an orange icon (indicating that this module can be used for clipping) in the object pool and the connection is indicated by a blue line

connecting the icons. At the same time, the graphics output generated by the *OrthoSlice* module becomes visible in the Viewer such as shown in Figure 5.7.

In the working area, there are various buttons, ports and sliders to allow a particular control parameter to be adjusted. Some of them are describe below:

Orientation: Provides three buttons for resetting the slice orientation. Axial slices are perpendicular to the z-axis, coronal slices are perpendicular to the y-axis, and sagittal slices are perpendicular to the x-axis.

Data Window: This port is displayed if linear mapping is selected. It allows the range of visible data values to be restricted. Values below the lower bound are mapped to black, values above the upper bound are mapped to white.

Slice Number: Allows the slice number to be changed via a slider.

Transparency: This radio box port determines the transparency of the slice. None means that the slices are fully opaque. Binary means that black parts are fully transparent while other parts are opaque. Alpha means that opacity is proportional to luminance. If a colormap is used for visualization opacity values are taken from there.

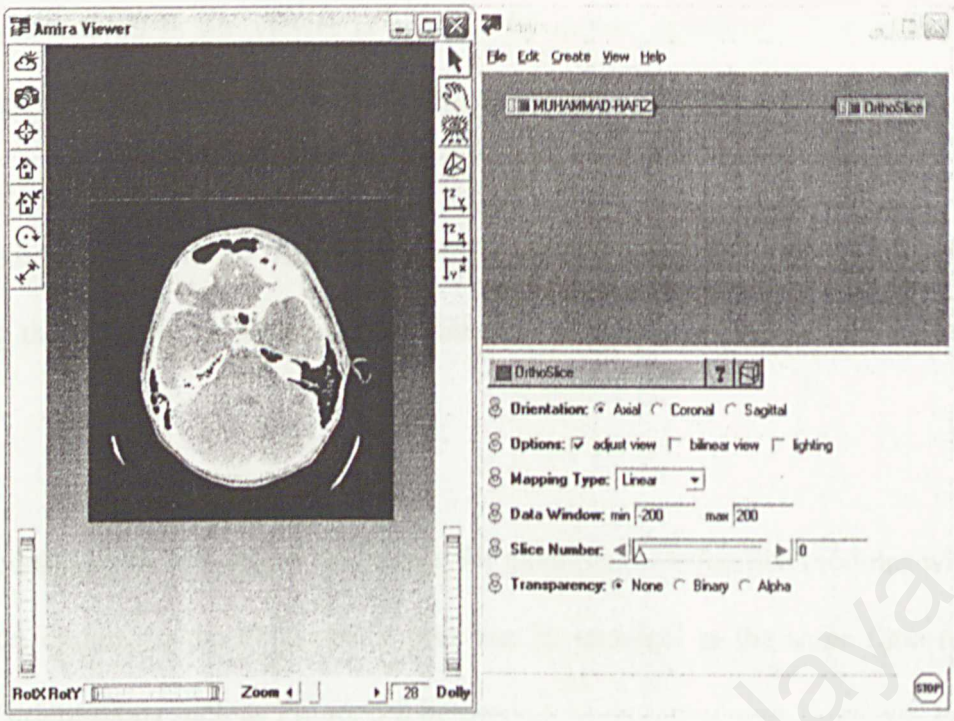


Figure 5.7: OrthoSlice Module

5.2.4 Interacting with the Viewer

The 3D viewer allows the model to be view from different positions. Moving the mouse inside the viewer window with the left mouse button pressed rotates the object. With the middle mouse button the object can be translated. For zooming press both the left and the middle mouse button at the same time and move the mouse up or down.

Inside the viewer window the mouse cursor has the shape of a little hand. This indicates that the viewer is in viewing mode. By pressing the "arrow" button (first button at the right side) the viewer is switched into interaction mode. In this mode, interaction with the

geometry displayed in the viewer is possible by mouse operations. For example, when using *OrthoSlice,* the slice number is changed by clicking on the slice and dragging it.

Each module has a *viewer toggle* by which the display can be switched off without removing the module. This button is attached to the colored bar where the module name is shown.

Same module can be attached a few times, for example 3 *OrthoSlice* modules with varied orientation chosen in the *Orientation* port can be attached at the same time to view 3 different orientations such as Figure 5.8 below:

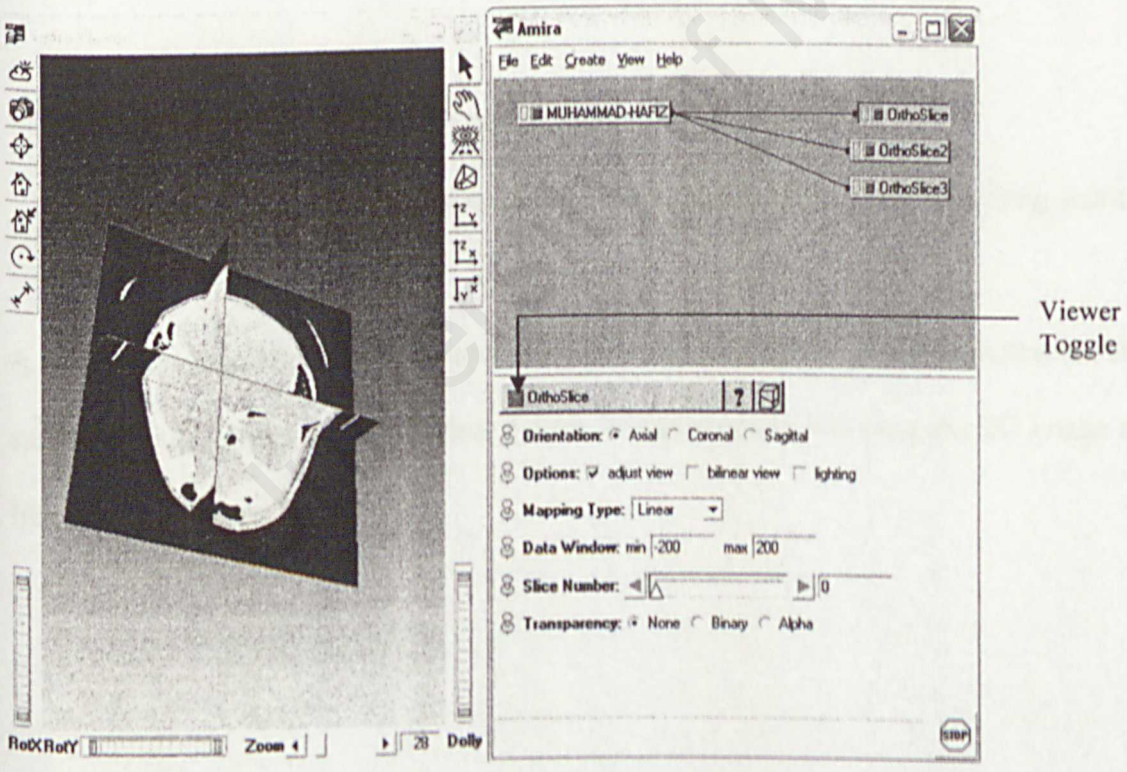


Figure 5.8: Image shown in 3 Different Orientations

To remove a module permanently, select it and choose *Remove* from the edit menu. Choose *Remove All* from the same menu to remove all modules.

5.3 Interactive Segmentation

Interactive (manual) segmentation assigns to each pixels of the image a 'Label' describing to which region or material the pixel belongs to (e.g., bone or kidney). The segmentation is stored in a separate data object called *LabelField*. Although this method requires more complicated steps compare to threshold segmentation (which will be discussed later) but it produces better result.

- Load the CT data file from the directory.
- Right click on the green icon and choose *LabelField* from the *Labelling* section.

A new green icon appears, the *LabelField* that will hold the segmentation results. At the same time, the ***Image Segmentation Editor*** Window opens, showing the 2D image slices like those in Figure 5.9.

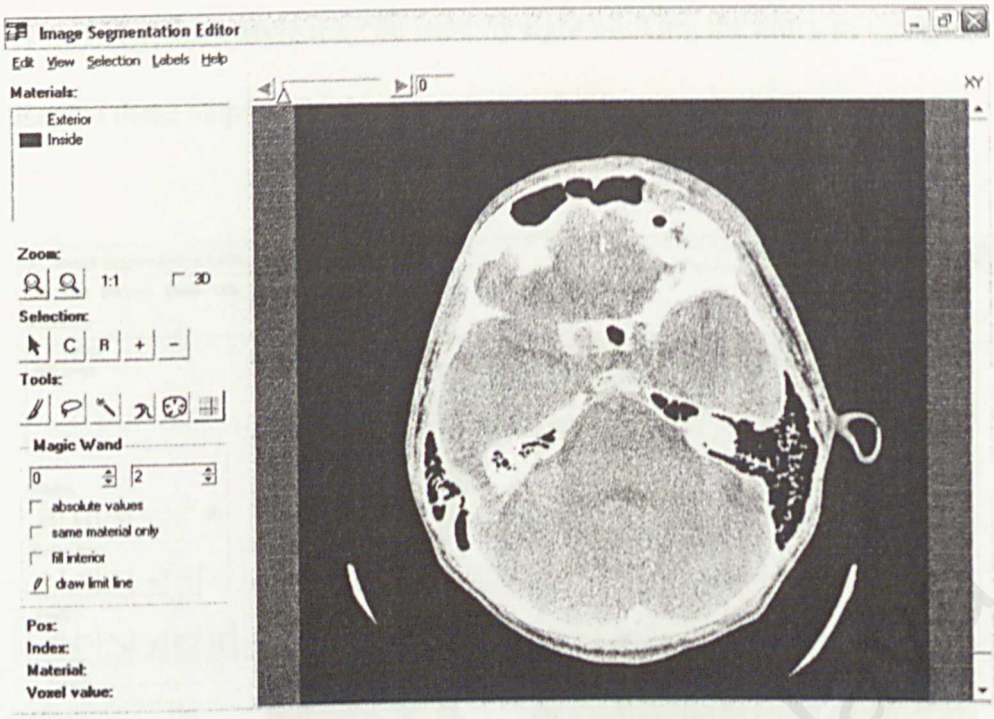


Figure 5.9: Segmentation Editor

Use the slider on the top to scroll through the slices. To select the desired region (bone or muscle) on the slices:

- Click on the *Brush* button under the label *Tools*.
- Mark the region with the mouse, selected pixels is displayed in red as shown in Figure 5.10.
- Hold down the control button to unselect wrongly selected pixels if necessary.
- Click into the *Material* list and choose *New Material* from the right button menu. Right click onto it to rename accordingly (e.g., bone or muscle).
- Hit the + button under the *Selection* label to assign the previously selected pixels to the material. Every pixel can only be assigned to one material.

- A different draw style may be used by right clicking the entry in the Materials.
- Repeat these steps on each region for each slice.

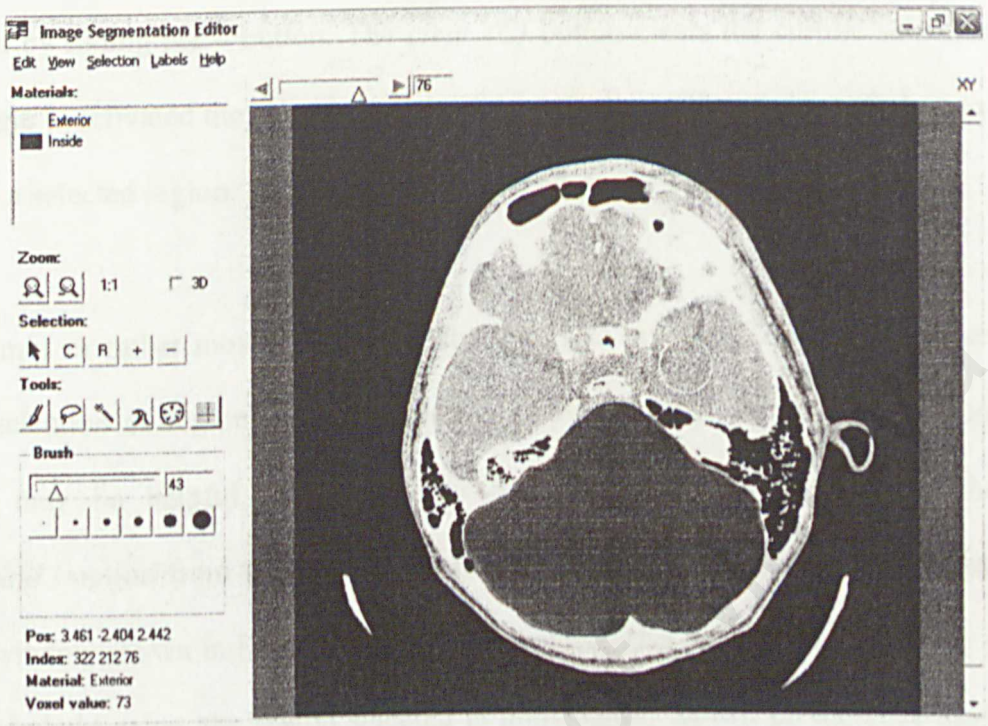


Figure 5.10: Selecting Region Using Brush Tool

If a structure does not change a lot from slice to slice, interpolation can be used:

- Go to slice 0 and mark the desired region using the brush.
- Go to a few slices ahead (e.g., slice 8) and mark the same structure.
- Choose from the menu bar: *Selection / Interpolate*.
- Scroll through the data set, the same region in between slice 1 to 7 are selected too.

In order to assign the selected pixels in all slices to the material, turn on the *3D toggle* near the *Zoom* buttons, select the material in the list, and click the + button. Then untoggle the 3D button again. Wrongly assigned region can be unassigned from the material by hitting the – button. The *clear (C)* button clears the current selection, if the 3D toggle is activated the selection is cleared in all slices. The *replace (R)* button tries to replace a selected region. Repeat the above procedure for subsequent slices.

Sometimes or rather most of the time, the data sets involved hundreds of slices which make selection of regions slice-per-slice very troublesome. In this case, the threshold option may be helpful. In the **Image Segmentation Editor** window, chose the *Threshold...*option from the *Selection* tool bar at the upper left corner. Subsequently, a small window shown in Figure 5.11 below appeared. Use the slider to select the min and max threshold value, the region selected is immediately shown on the slice itself. Click *OK* to close the window. If some other unwanted regions are also selected, they can be unselected by using the *Brush* tool while holding down the *Ctrl* button. Perform the same step to other slices. Each time the threshold window appears, the value shown is the one previously used. Therefore, it can be used again to select the same region which does not change much from the previous slice.

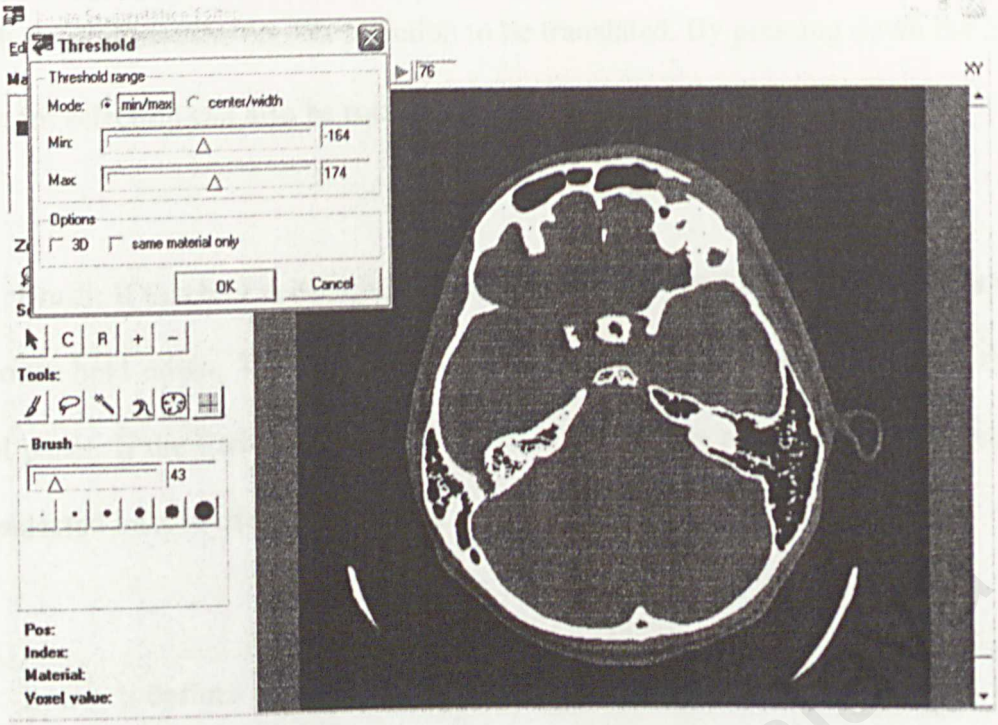


Figure 5.11: Selecting Region Using Threshold Value

The brush is only the most basic segmentation tool. The segmentation editor provides many more functions that are described below:

Pick & Move: This tool does two things. First, it selects a connected region assigned to one particular material by clicking onto an image voxel with the left mouse button. If the *select all* toggle is active, all voxels belonging to the same material as the clicked voxel will be selected, either in 3D or in the current slice only, depending on the value of the 3D toggle. If the *Ctrl* key is held down, voxels will be deselected. If the mouse is over an already selected pixel the mouse cursor takes on the shape of a hand.

Then the tool allows the current selection to be translated. By pressing down the *Shift* key the current selection can also be rotated.

 Brush: If this tool is activated, regions can be selected by painting voxels with the left mouse held down. The size of the brush can be modified via a slider in the tool's control panel. If the Ctrl key is held down or if the middle mouse button is used, pixels are deselected instead of being selected.

 Lasso: It defines an area by generating a closed contour curve. Usually, we draw such a curve freehand with the left mouse button pressed, but it is also possible to fence an area with line segments by pressing *Alt*, and clicking successively to points with the left mouse button (holding down the *Alt* key all the time). Successive points will be connected with straight line segments. To finish interaction release the *Alt* key and right click again. The contour is then closed and filled automatically.

 Magic Wand: This tool performs a so-called region growing either in 2D or in 3D, depending on whether the 3D toggle is activated or not. Clicking with the left mouse button on a voxel selects the largest connected area that contains the voxel itself and all voxels with gray values lying inside a user-defined range. The range can be specified via two spin boxes shown in the tool's control panel. The values of the spin boxes either define absolute gray values or values relative to the gray value of the seed pixel, depending on whether the absolute values toggle is activated or not. If necessary, the range will be automatically modified so that it contains the seed voxel. It is possible to

modify the range even after the seed voxel has been selected. For convenience, the lower threshold can be quickly modified by clicking the middle mouse button (or shift-clicking the right mouse button) and moving the mouse horizontally.

 Propagating Contour: This tool is a fast Active Contour. From a number of specified seed points a boundary front evolves. It computes the position of the front at all times up to the stop time. A slider and a text box allow the user select the appropriate front after pressing the initialization button. There are three buttons: Menu pops up the parameter menu, *Clear* removes the current seed points, and Init starts the computation.

 Blow tool: By clicking with the left mouse button on a voxel and dragging the mouse without releasing the button an initially circle-formed contour blows up. The greater the distance to the initial position of the mouse click gets the more the contour grows. The contour is designed to grow in areas with homogeneous grey values and to stop where grey values change abruptly such as the image edges. The tolerance slider in the option panel controls how sharp the image edge has to be in order to stop the contour from growing at each contour point. The smaller the tolerance the sharper the image edge has to be. If the tolerance is large, the contour will even stop at weak edges. After releasing the mouse button the area within the contour will be marked. Before the computation the image is smoothed using a Gaussian smoothing filter. The width of the filter can be changed within ranges of 1 (no smoothing) to 8 (very broad smoothing). The default value is 4.

 Crosshair: The crosshair tool is only active in 4 viewer mode. It simply displays a crosshair in all three orthogonal viewers. The colors denote the different directions (red=x-axis, green=y-axis, blue=z-axis). Clicking into one of the viewers moves the crosshair and simultaneously updates the two other viewers so that the crosshair's center is visible in all three viewers.

At this point, the segmentation process has finished using various tools. Save the label field to create a 3D surface model later, close the Image Segmentation Editor.

5.4 Threshold Segmentation

This is an alternative way of segmentation which can be achieved automatically purely based on the grey values of the image data set. As oppose to interactive segmentation, this method requires less manual interaction, but only works for images with good quality.

The first step is to separate the object from the background. This is done by segmenting the volume into exterior and interior regions based on voxel values.

- Load the CT data record from the directory, a green data icon appeared.
- Attach a *LabelVoxel* module by right clicking the green data icon and select it.
- A red data icon appeared, use the predefined threshold or type desired threshold value into the text field of each port.
- Press the *DoIt* button of Action Port.

By this procedure each voxel having a value lower than the threshold is assigned to the first region and each voxels whose value is greater than or equal to the threshold is assigned to the second region. Up to five different regions separated by four different thresholds can be extracted. For CT images the four regions *Exterior, Fat, Muscle,* and *Bone* are predefined such as shown in Figure 5.12 below. However, the name and color of these regions as well as the thresholds may be reset by the user. In order to find suitable thresholds, an image histogram showing the (absolute) number of voxel values and the current partitioning of the grey value range into the segments is provided by clicking on the *Histo* action button.

After pressing *DoIt* button a new data object (*.Labels*) with green icon appears in the Object Pool. It is of type *LabelField* which will hold the segmentation results.

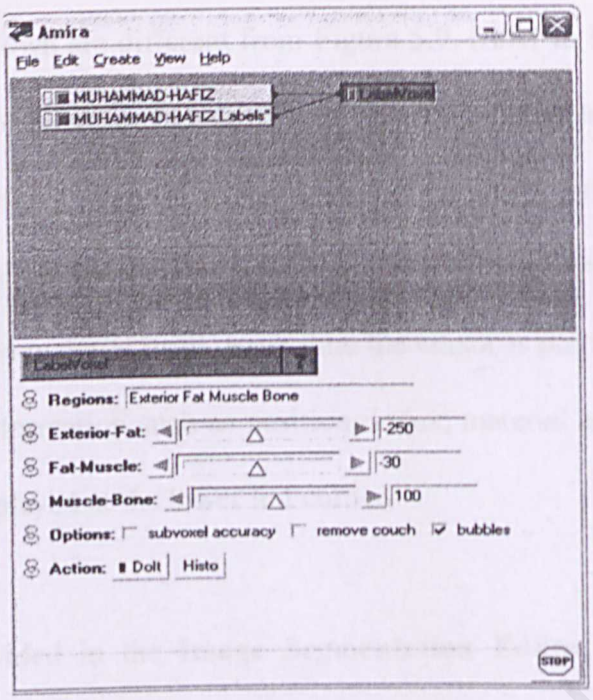


Figure 5.12: LabelVoxel Module

Sometimes or rather most of the time, threshold segmentation does not produce satisfying result. Some regions might not be assign properly, for example the skin might overlapped with fats, the couch the patient is lying on might be assigned to bone, the brain to the muscle and so on. This produces incorrect results, therefore refining is needed using the **Image Segmentation Editor**.

- Select *LabelField* data object by clicking on the green icon in the object pool.
- At the work space, click the 'pencil' icon button to invoke the **Image Segmentation Editor**.

This time the slices shown are different from Figure 5.9, (refer to Figure 5.13). Here the regions have been labeled according to materials define in module *LabelVoxel* in the previous steps (by pressing the *DoIt* button). Each material is marked by different color denoted in the Material list at the upper left corner, the color and draw style can be edited by right clicking on the material itself. Each time the cursor is pointed to the voxel of the slice in the viewer, information such as position, index, material and voxel value of that particular voxel is displayed at the lower left corner.

Using the tools provided in the **Image Segmentation Editor,** results from module *LabelVoxel* can be further refined. Image segmentation can be a time-consuming process especially when dealing with large data sets. Due to limited main-memory and for performance reasons, there is only a limited undo space for 2D interaction and most 3D operations cannot be undone. Therefore it is highly recommended to frequently save the label field during the process of segmentation.

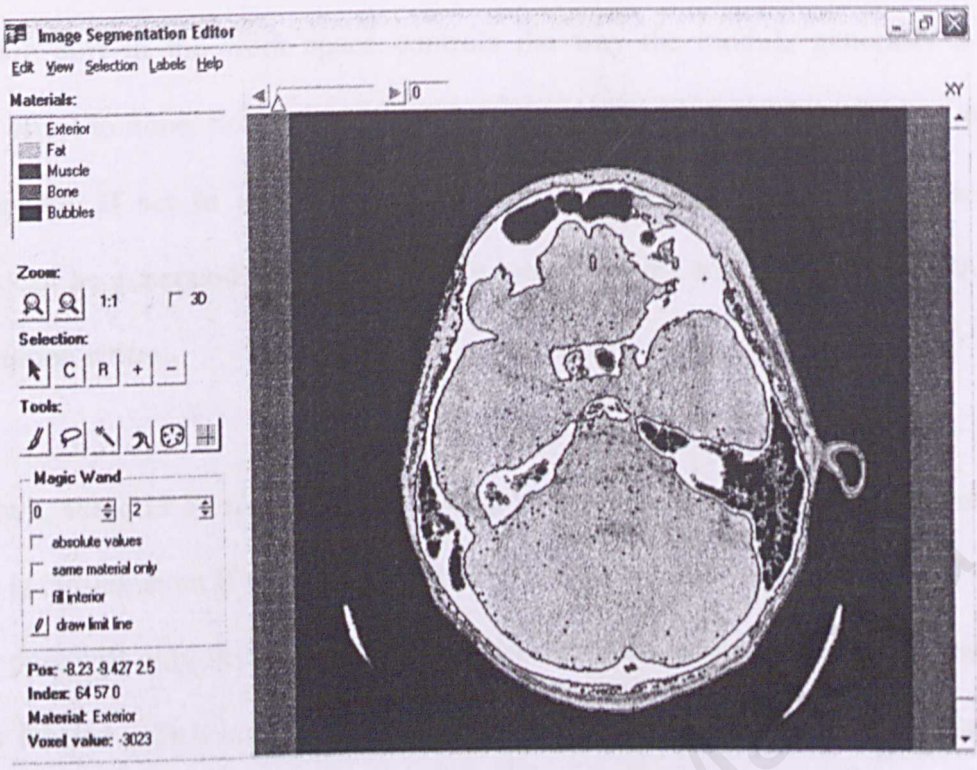


Figure 5.13: Image after Labeling

5.5 Constructing Surfaces from Segmentation Result

Now a triangular surface will be constructed by using the segmentation result from above. This surface is use for visualization later.

- Attach a *SurfaceGen* module to the *.Labels* data. A red data icon appeared.
- Press *Triangulate* in the Action Port.

This module (*SurfaceGen*) computes a triangular approximation of the interfaces between different tissue types in a *LabelField* with either uniform or stacked coordinates. The

Smoothing port in the work space controls the way the module generates a smooth surface. If set to none, no sub-voxel weights are used and the resulting surface will look staircase-like. If set to existing weights, then pre-computed weights are used; such weights can be generated with the resample module or the smoothing filter in the image segmentation editor.

By default, the *add border* in the *Options* port is turned on to ensures that the created surface is closed, even if some regions extend up to the boundary of the *LabelField*. The *Border* port will only be visible if *add border* has been selected. It provides two toggle buttons labeled *adjust coords* and *extra material*. It the first option is selected points belonging to triangles adjacent to boundary voxels will be moved exactly onto the nearest boundary face of the bounding box. In this way the resulting surface appears to be sharply cut off at the boundaries. The second toggle indicates that triangles adjacent to boundary voxels will be inserted into separate patches. The outer region of these patches will be called Exterior2. If the toggle is off no such extra material will be created. Instead, the boundary is assumed to be labeled with 0, which usually corresponds to Exterior.

In the *minimal edge length* port, a non-vanishing value indicates that short edges of the final surface should be contracted in order to increase triangle quality as well as to decrease the number of triangles. The value of the port indicates the minimal allowed edge length relative to the size of a unit grid cell. On default, values between 0 and 0.8 can be entered. Typically, a value of 0.4 already yields good results.

After pressing the *Triangulate* button (which require some computation time to generate the surface depending on the size of the data), a new green data object *.surf* appeared such as Figure 5.14 below.

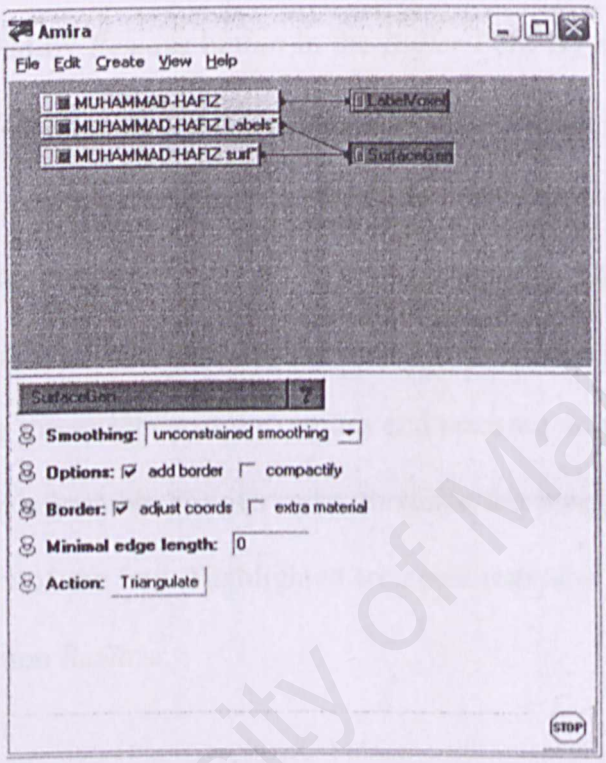


Figure 5.14: SurfaceGen Module

5.6 Visualizing Segmented Data

At this point, a surface has been generated to the segmented data and ready to be visualized.

- Attach a *SurfaceView* module to the *.surf* data object.

- A yellow icon appeared, followed by the 3D image in the *Viewer Screen*.

The 3D image showed is composed of different segment parts denoted by different colors shown in Figure 5.15 below. To view a particular segment, first clear the *Viewer* by pressing *Clear* followed by *Remove* button in the *Buffer* Port. By selecting two different regions in the *Materials* Port, all triangles separating these two regions are highlighted. Highlighted triangles are displayed in red wire-frame. By pressing button *Add* of port *Buffer* highlighted triangles can be added to the internal buffer, which causes them to be displayed in their own colors. Other segments may also be added on top of the previous segments by repeating the selection of the regions and pressing *Add*. Highlighting may be restricted by using an adjustable box. In order to resize the box, pick one of the green handles at the corners of the box. Highlighted triangles may also be removed from the buffer by pressing button *Remove*.

The 3D image may be viewed in different draw style such as outlined, shaded, lines, points, and transparent. These options are provided in the *Draw Style* port, is transparent is chosen, its level of transparency can be adjusted using the *Base trans* port.

The image can also be visualized in 4 different color modes provided in the *Colors* port:

- *Normal*: Each side of a triangle is colored according to the color of the opposite region.

- *Mixed*: Both sides of a triangle are colored in the same color, which is a mixture of the colors of the two sides in normal mode.
- *Twisted*: Each side of a triangle is colored according to the color of the adjacent region.
- *Boundary id*: Color denotes the boundary ids of the triangles. For each boundary id a separate color can be defined in the surface's parameter section. Boundary ids can be set and removed using the surface editor.

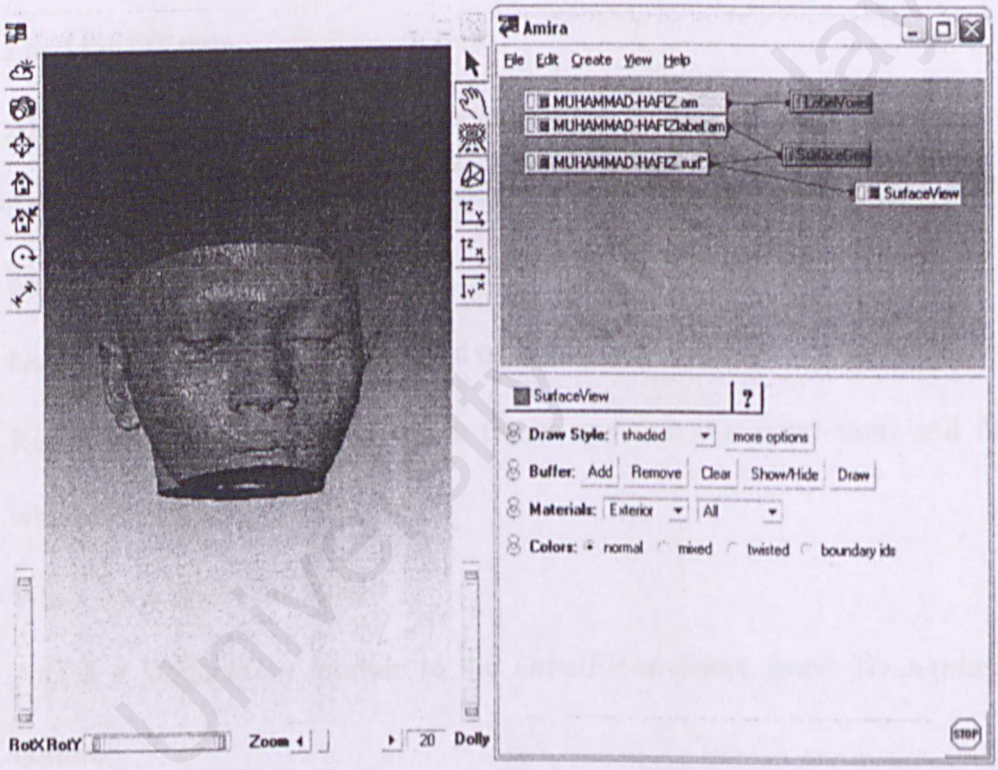


Figure 5.15: SurfaceView Module

5.7 Segmenting Real Life Data

The following CT data is taken from a patient who suffers from the growth of a tumor at her side cheek near the teeth bone. The image in the Figure 5.16 clearly shows the tumor with a darker shade has 'eaten' up the teeth bone.

The following steps are used to segment each part:

- Load the CT data.
- Attach a *LabelVoxel* module, press *DoIt* action button.
- Select *LabelField* object, press the 'pencil' icon to invoke the Image Segmentation editor.
- Using Brush tool, draw on the area of tumor and assign it to material *Tumor*.
- Repeat this for every slice where tumor appears; use other tools and functions when necessary.
- When done, close the editor.
- Attach a *SurfaceGen* module to the *LabelField* object, press *Triangulate* action button.
- When *.surf* data appears, attach *SurfaceView* module to it.
- Use the ports in *SurfaceView* to view various parts separately or together.
- Save the data and network created.

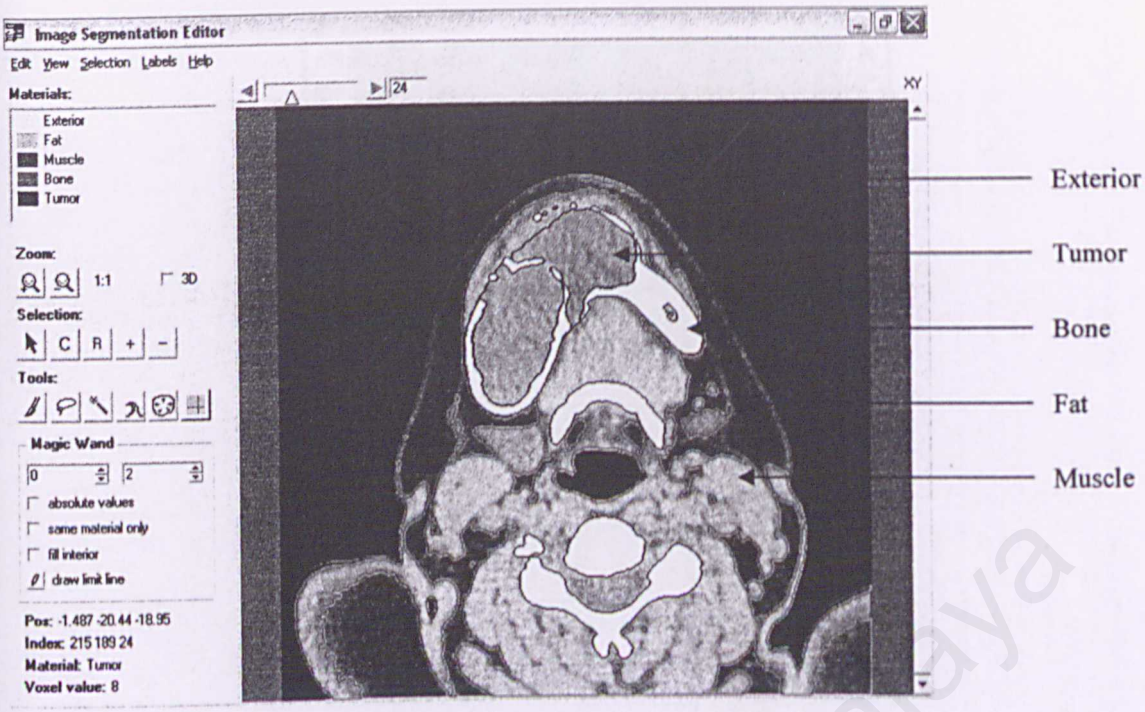


Figure 5.16: Data Sets with Tumor

The Figures 5.17 – 5.21 below show the segmentation result. To view only one particular material such as bone, remove existing image in the viewer by pressing *clear* follow by *remove* button. Select *Bone* and *All* in the material port. To view Bone and Tumor together, retain the Bone image in the viewer, select *Tumor* and *All* in the material port then press *Add* button.

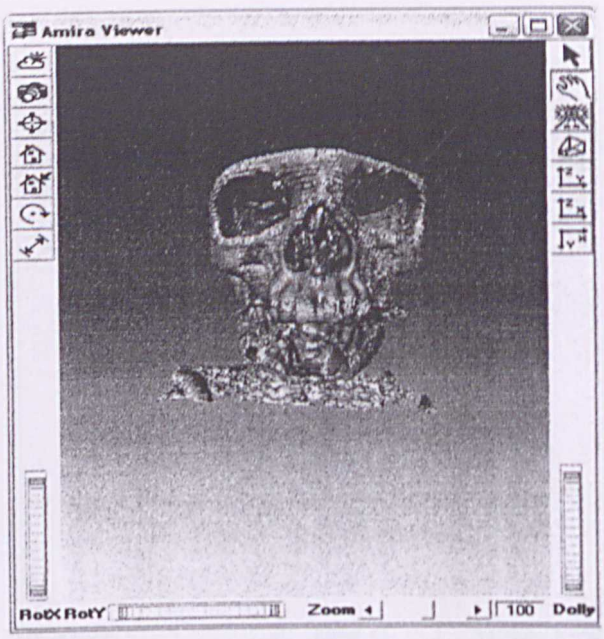


Figure 5.17: Segmented Bone

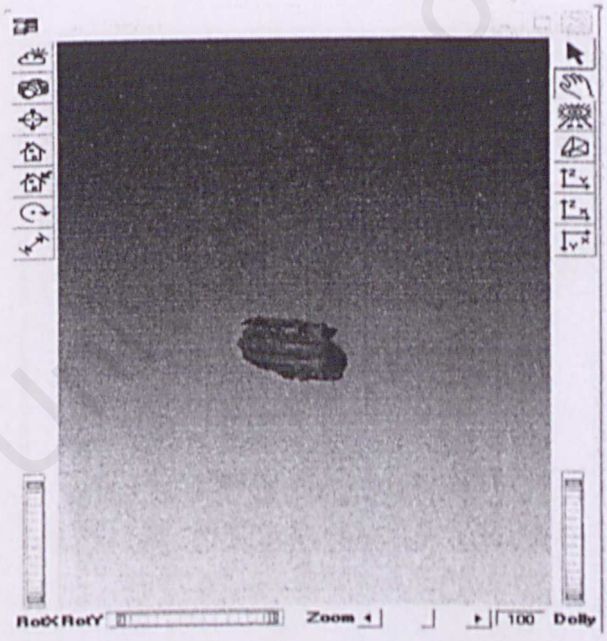


Figure 5.18: Segmented Tumor

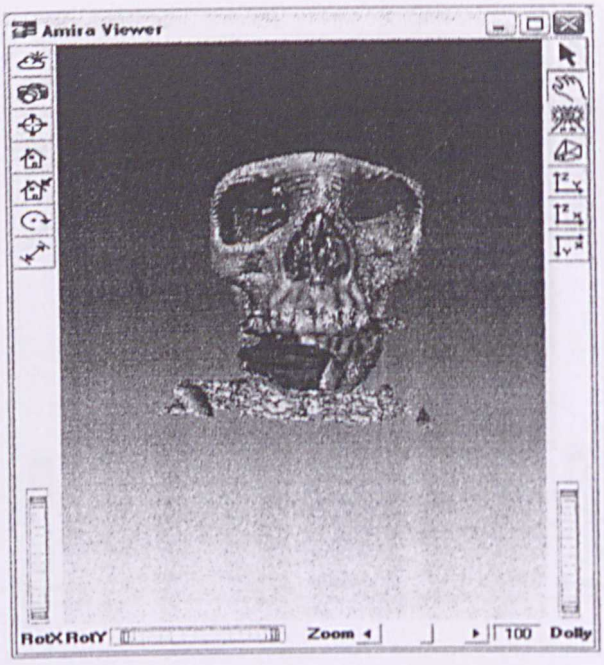


Figure 5.19: Segmented Bone and Tumor

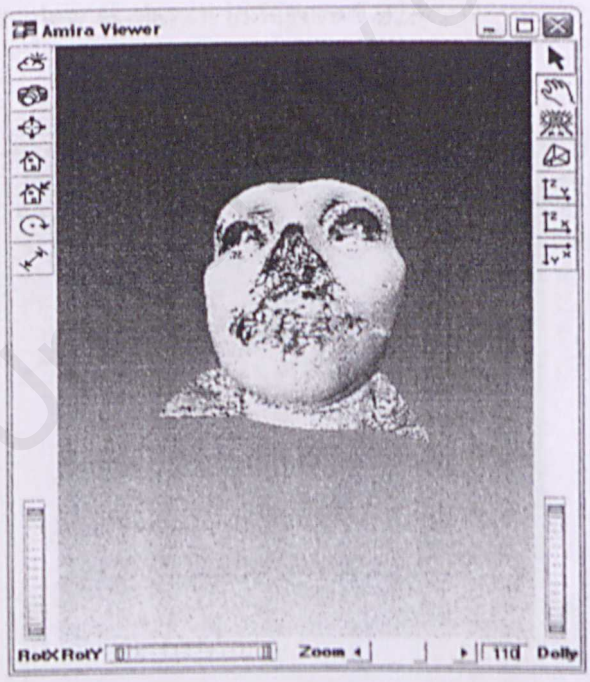


Figure 5.20: Segmented Fat

Figure 5.21: Segmented Muscle

To view the inner part of the image, use the Draw button to cut across the 'head' and rotate the image to side view as shown in Figure 5.22.

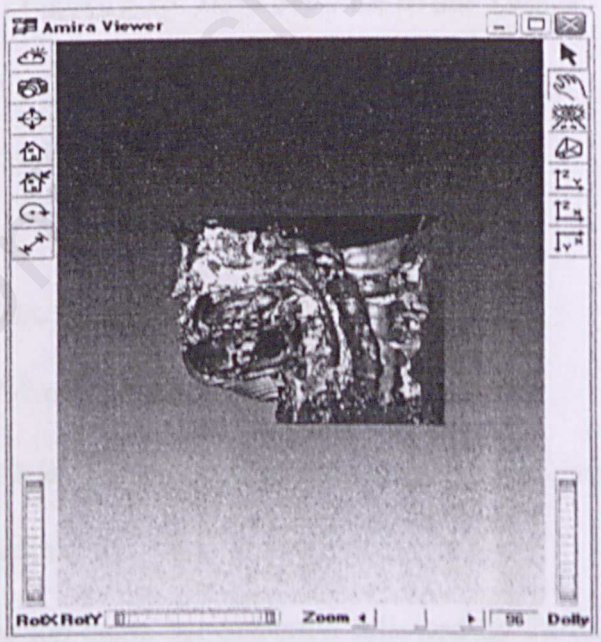


Figure 5.22: Side View of All Material

Next, the CT data taken is from a patient who has a brain tumor. From Figure 5.23, it is quite easy to spot the tumor as it has a much brighter shade compare to the brain.

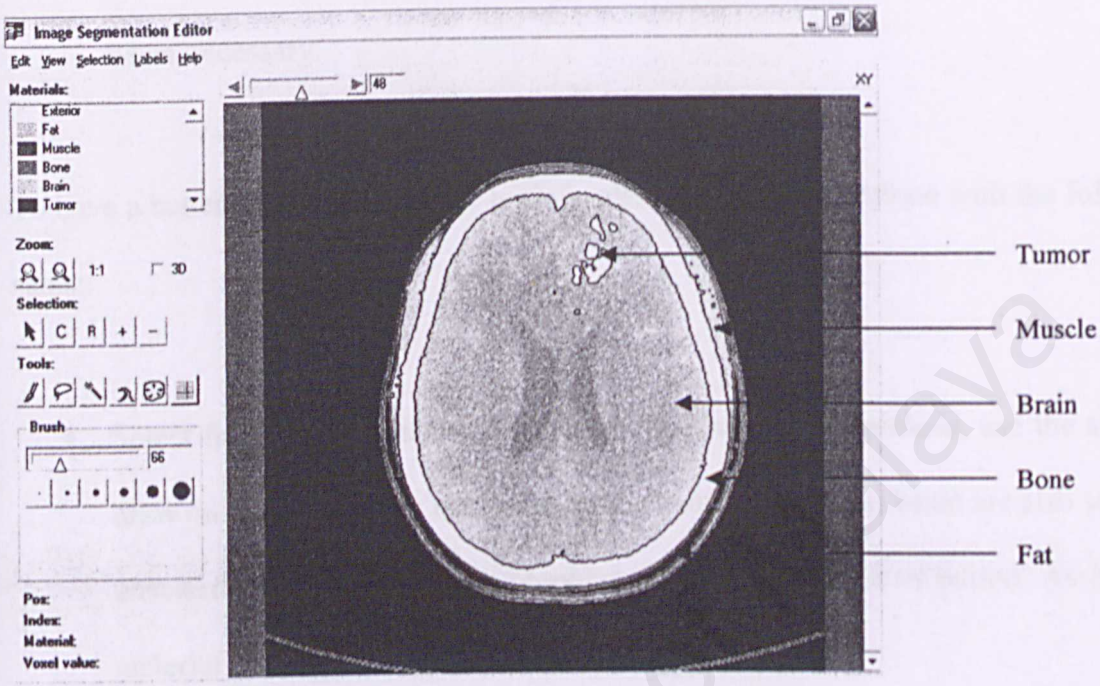


Figure 5.23: Data Set with Brain Tumor

To segment the tumor, follow the steps below:

- Load the CT data.
- Attach a *LabelVoxel* module, press *DoIt* action button.
- Select *LabelField* object, press the 'pencil' icon to invoke the Image Segmentation editor.
- Select from the top menu bar *Selection* followed by *Threshold*, use the slider to draw on the tumor. Note that other regions not belonging to tumor are also

selected; unselect it by using the *brush* tool while holding down *Ctrl* button. Assign it to material *Tumor*.

- Repeat this for every slice where tumor appears; use other tools and functions when necessary.

To have a better view of tumor, the brain is also segmented. Continue with the following steps:

- Select from the top menu bar *Selection* followed by *Threshold*, use the slider to draw on the brain. Note that other regions not belonging to brain are also selected; unselect it by using the *brush* tool while holding down *Ctrl* button. Assign it to material *Tumor*.
- Repeat this for every slice where brain appears; use other tools and functions when necessary.
- When done, close the editor.
- Attach a *SurfaceGen* module to the *LabelField* object, press *Triangulate* action button.
- When *.surf* data appears, attach *SurfaceView* module to it.
- Use the ports in *SurfaceView* to view various parts separately or together.
- Save the data and network created.

The results of segmented parts are shown from Figure 5.24 – 5.27.

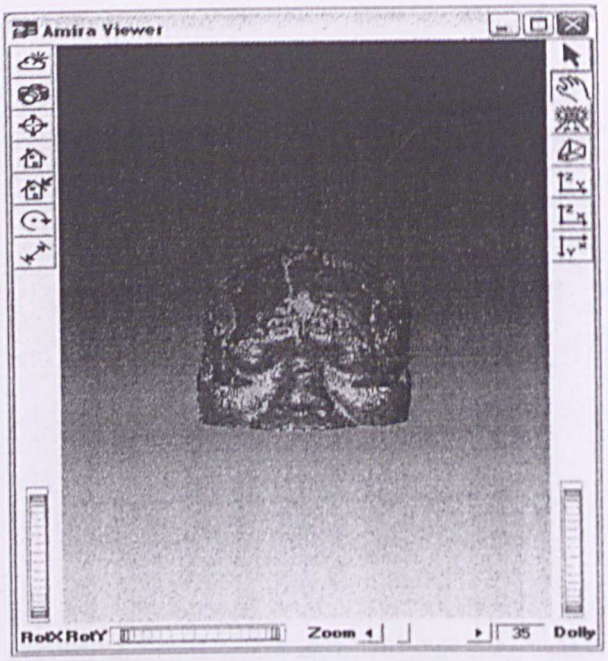


Figure 5.24: Segmented Muscle

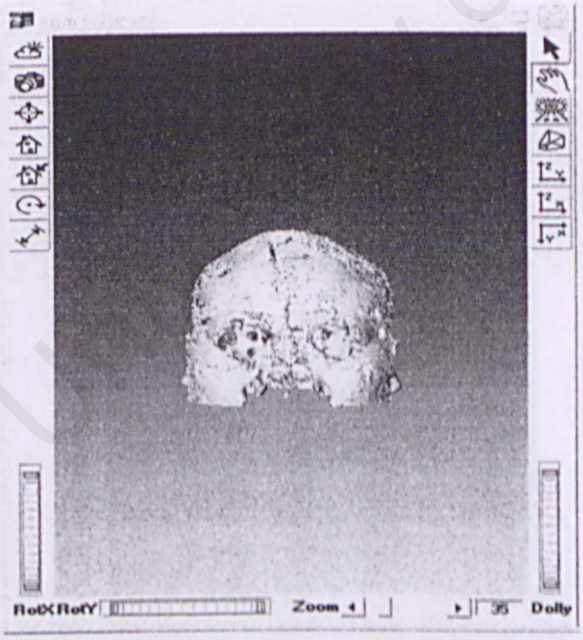


Figure 5.25: Segmented Fat

Figure 5.26: Segmented Bone

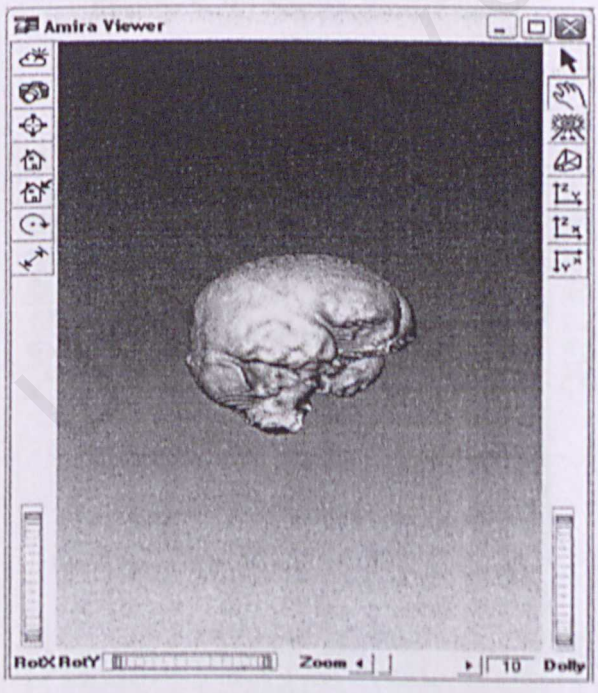


Figure 5.27: Segmented Brain

To view the tumor inside the brain such as Figure 5.29, the *Draw Style* port in *SurfaveView* module can be set to transparent mode.

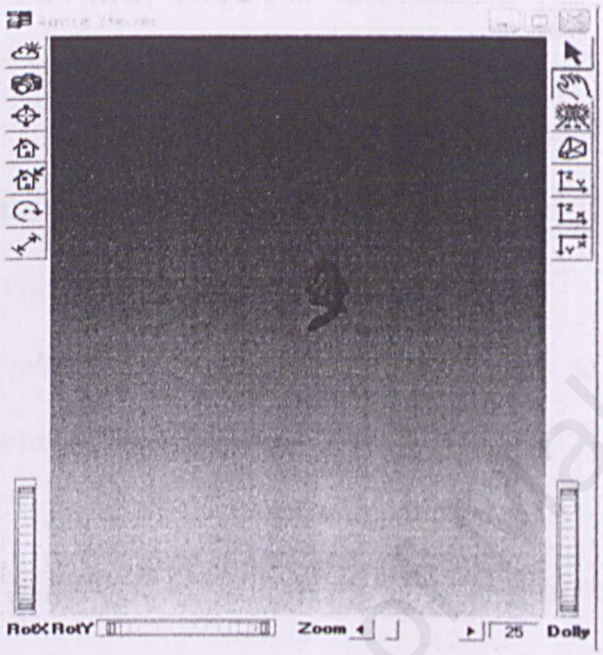


Figure 5.28: Segmented Tumor

Figure 5.29: Segmented Brain and Tumor

Finally, the CT data is taken from a patient who had a fractured bone near the eye socket. This is clearly shown in Figure 5.30.

The following steps are used to segment each part:

- Load the CT data.
- Attach a *LabelVoxel* module, press *DoIt* action button.
- Select *LabelField* object, press the 'pencil' icon to invoke the Image Segmentation editor.

Note that the couch where the patient is lying on is assigned to material bone thus produces wrong result. To remove or assign couch to Exterior, continue with the steps below:

- Using the *brush*, draw on couch for each slice or use the interpolation method.
- Assign it to Exterior.
- When done, close the editor.
- Attach a *SurfaceGen* module to the *LabelField* object, press *Triangulate* action button.
- When *.surf* data appears, attach *SurfaceView* module to it.
- Use the ports in *SurfaceView* to view various parts separately or together.
- Save the data and network created.

Figure 5.30: Data Set with Fractured Bone

The following Figures 5.31 – 5.33 show the segmentation results.

Figure 5.31: Segmented Muscle

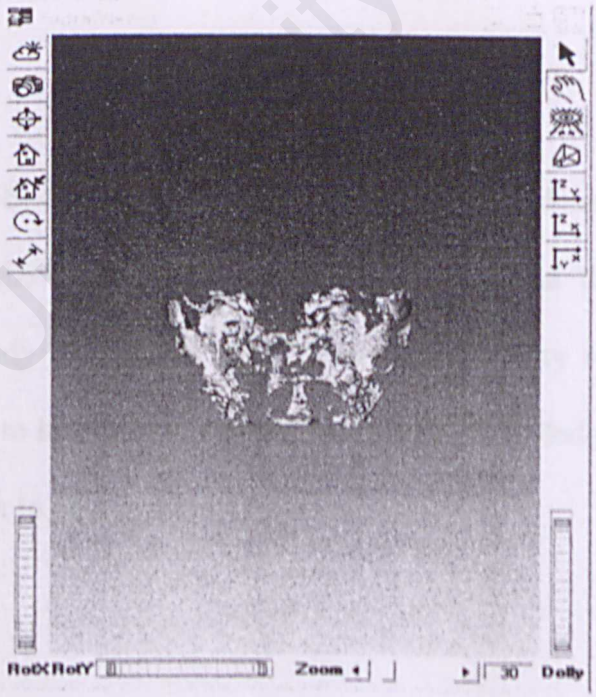


Figure: 5.32: Segmented Fat

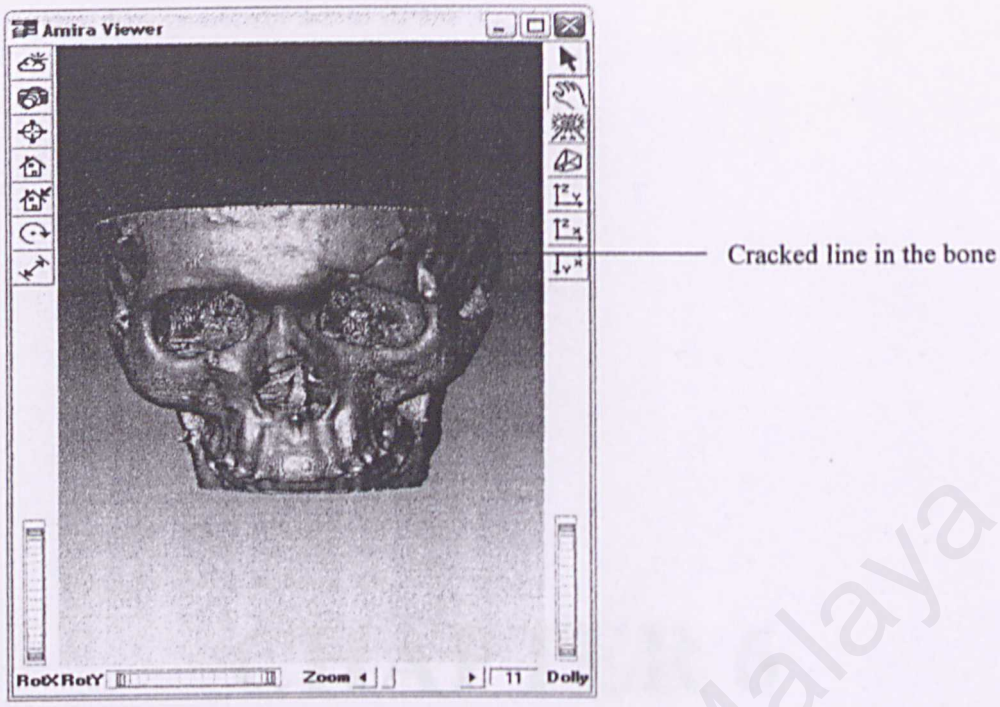


Figure 5.33: Segmented Bone

5.8 Summary

This chapter is a step-by-step tutorial on how segmentation is done and ways to visualize its results. Overall, the combination of both interactive and threshold segmentation together with its handy tools gives considerably good-quality results. However, it is highly recommended to have some background medical knowledge or seek advice from the experts for in order to master segmentation skills.

CHAPTER 6

DISCUSSION AND CONCLUSION

CHAPTER 6 – DISCUSSION AND CONCLUSION

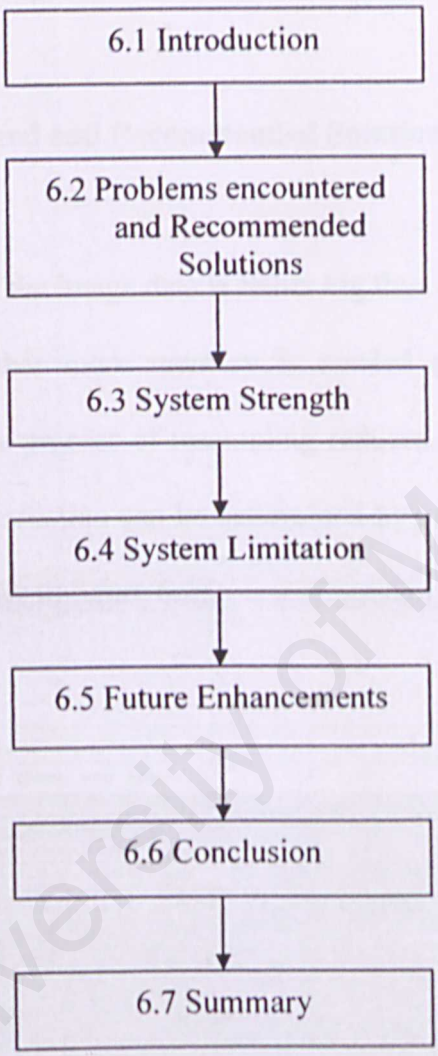


Figure 6.1: Overview of Chapter 6

6.1 Introduction

During the accomplishment of this project, unforeseen circumstances aroused thus hinder the process of implementation. Solution and decision need to be sought and made based

on specific problem tasks. Although the overall performance is satisfying, enhancements are needed to achieve a better system. The discussion on these issues as well as findings on system strengths and limitations will be covered in the following topics.

6.2 Problems encountered and Recommended Solutions

Most of the time, the size of the image data is rather big thus consumes too much time for computation. Therefore either more memory is needed (at least 512 MB) or else Resampling is required. The process of resampling reduces dimensions or voxel size of the image. The amount of reduction can be determined by using the *Mode* port provided in the *Resample* module (refer to figure 6.2).

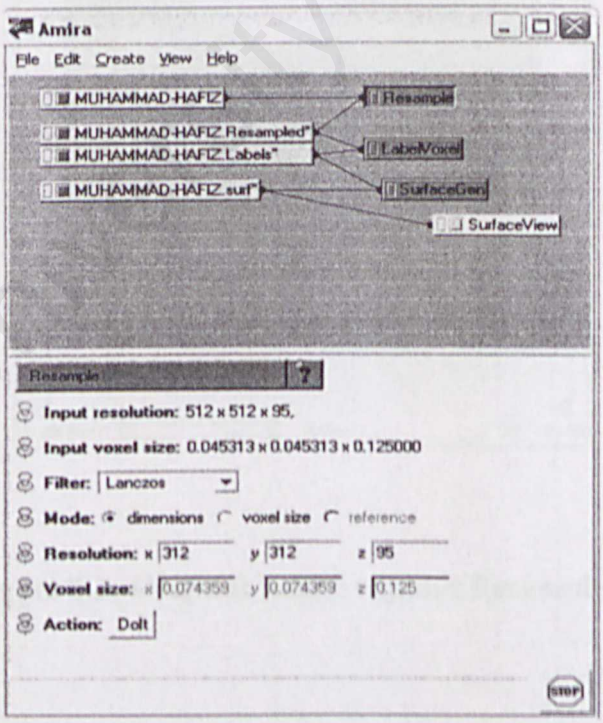


Figure 6.2: Resample Module

As a trade off, the image loses its quality and some detail information. A comparison between the same data before and after resampling is shown in the figures below. Figure 6.3 is the original data before resampling, its resolution and voxel size for X-Y-Z axis are 512×512×95 and 0.0453125×0.0453125×0.125. Figure 6.4 is the data after resampling, its resolution has been reduced to 312×312×95 with the voxel size of 0.074359×0.074359×0.125. It clearly shows that data which had been resampled produced a blurring effect, as opposed to the sharper image before resampling.

Figure 6.3: Original Image without Resampling

Figure 6.4: Image after Resampling

CT data with higher resolution produces better visualization result but consumes larger amount of storage and memory. Its resolution had been preset during the process of scanning; therefore it can only be down-sampled and not up-sampled. As a result, low resolution CT data produces poor images with little details. For example the brain has many ridges and grooves on its surface, to get them clearly seen after segmentation, the data must have high resolution.

Due to the lack of bio-medical knowledge towards human anatomy, it is extremely difficult to determine which region in the 2D slices belongs to which part of the body. This is major obstacle encountered during image segmentation. Sometimes the brain and the muscle can have same threshold value; a tumor can be mistakenly regarded as part of

the internal organ; a fractured bone is overlooked; a metal filling can distort the clarity of surrounding regions, etc. On top of that, some organs are not individual stand alone part that can be easily recognized; it could extend further or overlapped with other organ. For example the brain consists of a few lobes: Frontal, Parietal, Occipital, and Temporal which are extensions from the spinal cord. All these phenomenon complicate the segmentation process, thus careful planning and thorough research as well as advice from expertise are needed to produce a fine and accurate result.

6.3 System Strength

Amira supports various types of CT or MRI images stored in different formats such as DICOM, JPEG, STL and VRML. Upon loading the data, the slices are automatically arranged in stacked or uniform coordinates. Segmentation can be done not only in 2D mode but also in 3D mode. There are various useful tools and functions provided in the segmentation editor that ease the process of interactive segmentation.

For threshold segmentation, the selected region can be view immediately as the value is changed using the slider. Further more, the threshold values for exterior, bone, muscle and fat have already been predefined. Each segmented part can be viewed together or separately either in shaded or transparent mode. Next, the data can be saved individually or the whole network can be saved. Finally, the segmentation result can be exported as STL file (a CAD format for Rapid Prototyping); it is a faceted surface representation.

6.4 System Limitation

When dealing with large data sets, more memory is needed (preferably 512MB). If the memory is low, computation takes a very long time especially when for 3D image. Sometimes, 3D surface cannot be generated at all if the system could not allocate enough memory. Although the data sets can be down-sampled to adapt to memory limitation, but it loses detail information thus produces a less sharpen image.

Predefined threshold values for bone, muscle and fat might not be accurate. For example, the brain has same threshold value as muscle, thus it is assigned to material muscle.

6.5 Future Enhancements

More regions can be predefined depending on which part of the human body is taken. For example if the CT data examined is human head, then it would be better if the threshold value for brain can be predefined. Further more, it is recommended that Amira be able to detect the possibility of unusual occurrences such as tumor by comparing to a normal CT data of a healthy human being.

6.6 Conclusion

The project has met its objective in segmenting CT data despite having numerous problems and limitations. Overall, Amira is a user friendly 3D image visualization and

segmentation software. It provides multiple choices of segmentation tools and techniques thus successfully achieving its purpose as a useful segmentation tool for 3D CT data.

Throughout the development process, valuable knowledge was gained from the complexities and intricacies of Amira. Facts and information have been uncovered to further understand its functionalities. Last but not least, priceless experience and useful skills was learnt. Among them are problem solving skills, time management and mutual coordination.

6.7 Summary

This chapter presents a thorough discussion on the difficulties faced and ways to solve them. It also includes the strength and limitations of Amira in correspondence to 3D CT data segmentation. Next is the suggestion on future enhancement to produce a more refined and accurate segmentation result. Finally, a conclusion on the overall performance of Amira is made.

REFERENCE

[1] Scott E Umbaugh, *Computer Vision and Image Processing: A Practical Approach Using CVIP tools*. Upper Saddle River: Prentice Hall PTR, 1999

[2] Stephen R. Schach *Object Oriented and Classical Software Engineering*. New York, Americas: McGraw-Hill, 2004

[3] Aaron E. Lefohn, Joshua E. Cates, and Ross T. Whitaker. *Interactive, GPU-Based Level Sets for 3D Segmentation.* <http://graphics.cs.ucdavis.edu/~lefohn/work/rls/tumorSeg/gpuTumorSeg.pdf> (19 August 2004).

[4] Dzung L. Pham, Chenyang Xu and Jerry L. Prince. *A Survey of Current Methods in Medical Image Segmentation.* <http://medic.rad.jhmi.edu/dpham/p124r.pdf> (23 August 2004).

[5] James E. Cabral Jr. *ARIES: Software for Interactive Segmentation of Medical Images Featuring 3D Region Growing.* <http://icsl.ee.washington.edu/~cabralje/papers/thesis/section1.pdf> (23 August 2004).

[6] Kaushal Mehta. *A Practical Guide to the 3D Slicer: An Introductory Users Manual.* February 2001. <http://spl.bwh.harvard.edu:8000/pages/papers/slicer/manual/slicer_manual.htm> (29 August 2004).

[7] Ken Museth, David E. Breen, Leonid Zhukov and Ross T. Whitaker. *Level Set Segmentation From Multiple Non-Uniform Volume Datasets.* <http://www.gg.caltech.edu/~zhukov/papers/vis02-2.pdf > (17 August 2004).

[8] Mark Hasegawa-Johnson, Jul Setsu Cha, and Katherine Haker. *CTMRedit: a Matlab-Based Tool for Segmenting and Interpolating MRI and CT Images in Three Orthogonal Planes.* <http://www.ifp.uiuc.edu/speech/pubs/1999/embs99.pdf> (23 August 2004).

[9] Molly M. Dickens, Shaun S. Gleason and Hamed Sari-Sarraf. *Volumetric Segmentation via 3D Active Shape Models.* <http://www.ornl.gov/sci/ismv/pdfs/publications/gleason_volumetric%20segmentation%20via%203d.pdf> (19 August 2004).

[10] Morten Bro-Nielsen. *Mvox: Interactive 2-4D Medical Image and Graphics Visualization Software.* <http://www.mortenbronielsen.net/documents/mvox_submit.pdf> (29 August 2004).

[11] M. R. STYTZ, G. Frieder and 0. Frieder. *Three-Dimensional Medical Imaging: Algorithms and Computer Systems.* <http://delivery.acm.org/10.1145/130000/125155/p421-stytz.pdf> (17 August 2004).

[12] Raisa Z. Freidlin, Chikai J. Ohazama, Andrew E. Arai, Delia P. McGarry, Julio A. Panza and Benes L. Trus. *NIHmagic: 3D Visualization, Registration and Segmentation Tool.* <http://dcb.cit.nih.gov/~raisa/NIHmagic.pdf> (23 August 2004).

[13] Sarang Lakare. *3D Segmentation Techniques for Medical Volumes.* <http://www.cs.sunysb.edu/~mueller/teaching/cse616/sarangRPE.pdf> (19 August 2004).

[14] Sarang Lakare, Ming Wan, Mie Sato, and Arie Kaufman. *3D Digital Cleansing Using Segmentation Rays.* <http://sled.sourceforge.net/pub/papers/SegRaysVIS2000.pdf> (17 August 2004).

[15] *3D-DOCTOR: CREATE MORE ACCURATE 3D MODELS USING VECTOR-BASED TECHNOLOGIES.* <http://www.ablesw.com/3d-doctor/segment.html> (25 September 2004).

[16] *3DVIEWNIX.*

<http://biocomp.stanford.edu/3dreconstruction/software/3dviewnix.html>

(29 August 2004).